

原书第3版

人工智能 智能系统指南

(澳) Michael Negnevitsky 著 陈薇 等译

Artificial Intelligence
A Guide to Intelligent Systems Third Edition



计算机科学丛书

人工智能

智能系统指南（原书第3版）

**Artificial Intelligence: A Guide to
Intelligent Systems, Third Edition**

（澳）Michael Negnevitsky 著

陈 薇 等译

HZ BOOKS
华章图书



机械工业出版社

China Machine Press

www.cmbp.com 让未来触手可及

本书是一本很好的人工智能入门书籍，内容丰富、浅显易懂。作者根据自己多年的教学、实践经验，并结合实际代码、图示、案例等讲解了人工智能的基本知识。

全书共分 10 章，主要内容包括：基于规则的专家系统、不确定性管理技术、模糊专家系统、基于框架的专家系统、人工神经网络、进化计算、混合智能系统、知识工程、数据挖掘等。另外，本书还提供了一个人工智能相关术语表和包含商业化的人工智能工具的附录。

本书既可以作为计算机科学相关专业本科生的入门教材，也可以作为非计算机专业读者的自学参考书。

Michael Negnevitsky: Artificial Intelligence: A Guide to Intelligent Systems, Third Edition (ISBN 978-1-4082-2574-5).

Copyright © 2011 by Pearson Education Limited.

This translation of Artificial Intelligence: A Guide to Intelligent Systems, Third Edition (ISBN 978-1-4082-2574-5) is published by arrangement with Pearson Education Limited.

All rights reserved.

本书中文简体字版由英国 Pearson Education 培生教育出版集团授权出版。

封底无防伪标均为盗版

版权所有，侵权必究

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2011-4255

图书在版编目(CIP)数据

人工智能：智能系统指南（原书第3版）／（澳）尼格尼维斯基（Negnevitsky, M.）著；陈薇等译．—北京：机械工业出版社，2012.7

（计算机科学丛书）

书名原文：Artificial Intelligence: A Guide to Intelligent Systems Third Edition

ISBN 978-7-111-38455-7

I. 人… II. ①尼… ②陈… III. 人工智能 IV. TP18

中国版本图书馆 CIP 数据核字（2012）第 104490 号

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）

责任编辑：高婧雅

印刷

2012 年 8 月第 1 版第 1 次印刷

185mm×260mm·20.75 印张

标准书号：ISBN 978-7-111-38455-7

定价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991；88361066

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com 让未来触手可及

人工智能（Artificial Intelligence, AI）是计算机学科的一个分支，被认为是 21 世纪三大尖端技术（基因工程、纳米科学、人工智能）之一。“人工智能”一词最初是在 1956 年达特茅斯学院研讨会上提出的。从那以后，研究者们发展了众多理论和原理，人工智能的概念也随之扩展。在经历了辉煌和低谷之后，近 30 年来，人工智能的研究取得了迅速的发展，在很多领域（例如工程、医疗、财经、商业和管理等）都获得了广泛应用。人工智能的研究逐渐成熟，已成为一个独立的分支，无论在理论和实践上都已自成一个系统。

本书作者 Michael Negnevitsky 是澳大利亚塔斯马尼亚大学电气工程和计算机科学系教授。他的许多研究课题都涉及人工智能和软计算，一直致力于电气工程、过程控制和环境工程中智能系统的开发和应用。他著有 200 多篇论文、两本书，并获得了四项发明专利。

本书是一本很好的人工智能入门书籍，内容丰富、浅显易懂、适应面广。美国、德国、日本等多所大学的计算机相关专业都采用本书作为教材或主要教学参考书。与第 2 版相比，第 3 版引入了关于“数据挖掘”的新的一章，并展示了智能工具在解决复杂实际问题中的新应用。第 3 版还扩展了本书的参考文献和参考书目，并更新了附录中的人工智能工具和厂商列表。通过本书的学习，相信读者一定会对人工智能的完整知识体系有全面的了解。

陈薇组织并参与了本书的翻译和审校工作，参加翻译工作的还有闫秋玲、黄威靖、欧高炎、刘璐。由于译者水平有限，译文中疏漏和错误之处在所难免，欢迎广大读者批评指正。

HZ BOOKS
华章图书

陈 薇

2012 年于北京大学

第3版前言

Artificial Intelligence, 3E

这本书的主要目的与第1版相同，即为读者提供一本能实际了解计算机智能领域相关知识的书。它适合作为一个学期课程的入门教程，学生需要具备一些微积分的知识，不要求具备编程的经验。

在涵盖内容方面，本书引入了关于数据挖掘的新的一章，并展示了智能工具在解决复杂实际问题中的新应用。主要的变化如下：

在新的“数据挖掘和知识发现”一章中，我们介绍了大型数据库中知识发现不可或缺的一部分——数据挖掘。涉及将数据转换为知识的主要技术和工具，包括统计方法、数据可视化工具、结构化查询语言、决策树和购物篮分析。同时还展示了几个数据挖掘应用的实例。

在第9章中，增加了采用自组织神经网络进行聚类分析的新的实例。

最后，我们还扩展了本书的参考文献和参考书目，更新了附录中的人工智能工具和厂商列表。

Michael Negnevitsky

2010年9月于澳大利亚
塔斯马尼亚州 霍巴特市

“The only way not to succeed is not to try.”

——Edward Teller

又是一本人工智能的书……我已经见过很多同类的书，为什么还要理会它？它有什么与众不同之处？

每年，有成百上千本书和博士论文拓展着计算机或人工智能的知识体系。专家系统、人工神经网络、模糊系统以及进化计算是应用于智能系统的主要技术，数百个工具支持着这些技术，数以千计的科学论文不断推进着该学科的发展。本书中的任何章节的内容都可以作为一本书的主题。然而，我想写一本能够阐述智能系统基础的书，更为重要的原因是，我想消除大家对人工智能理论的畏惧心理。

大多数人工智能文献是采用计算机科学的专业术语进行描述的，其中充斥着大量复杂的矩阵代数和微分方程，这当然给人工智能理论带来了令人敬佩的资本，但同时也令非计算机科学的科学家对其敬而远之。不过，这种情况已经有所改变！

个人电脑已经成为我们日常生活中不可或缺的部分，我们将它用于打字机和计算器、日历和通信系统、交互式数据库以及决策支持系统。并且我们还渴望更多。我们希望计算机智能化！我们发现智能系统正快速地走出实验室，而我们也想更好地利用它。

智能系统的原理是什么？它是如何构建的？智能系统的用处是什么？我们该如何选择适当的工具构建智能系统？这些问题都可以在本书中找到答案。

与许多介绍计算机智能的书不同，本书将智能系统背后的奥妙简单明了地展示给读者。它是基于作者多年来给没有多少微积分知识的学生授课时所用的讲义而编写的，读者甚至不用学习任何编程语言就可轻松理解！书中的素材已经经过笔者 15 年的教学实践的检验，写作中也考虑了学生们提出的典型问题和建议。

本书是一本计算机智能领域的入门书籍，内容包括基于规则的专家系统、模糊专家系统、基于框架的专家系统、人工神经网络、进化计算、混合智能系统、知识工程和数据挖掘。

总体来说，本书可作为计算机科学、计算机信息系统和工程专业的本科生的入门教材。我在教学过程中，会要求学生开发小型的基于规则和基于框架的专家系统，设计一个模糊系统，探究人工神经网络，采用遗传算法求解一个简单的优化问题，并开发混合的神经模糊系统。他们使用一些专家系统的核心程序（XpertRule、Exsys Corvid 和 Visual Rule Studio）、MATLAB 的模糊逻辑工具箱以及 MATLAB 的神经网络工具箱。我们选择这些工具的原因是它们能够便利地演示教学中的原理。然而，本书并不局限于任何特定的工具，书中给出的例子可以轻松地在不同的工具中实现。

www.aibbt.com 让未来触手可及

本书也适合非计算机专业的相关人士自学。对于他们来说，本书提供了进入基于知识

的系统和计算智能的前沿领域的钥匙。事实上，本书面向的专业读者群十分广泛：工程师和科学家、管理人员和商人、医生和律师，也就是所有面临挑战而无法用传统的方法解决问题的人，所有想了解计算机智能领域巨大成就的人。本书将帮助你实际了解智能系统的用途，发现与你的工作密切相关的工具，并最终学会如何使用这些工具。

希望读者能与我共同分享人工智能和软计算学科所带来的乐趣，并从本书中获益。

读者可以访问 <http://www.booksites.net/negnevitsky> 获得更多的信息。

Michael Negnevitsky

2001 年 2 月于澳大利亚

塔斯马尼亚州 霍巴特市



本书一共包含 10 章。

第 1 章简要介绍了人工智能的历史，从 20 世纪中期诞生人工智能的思想并在 60 年代设立了远大目标并积极实现到 20 世纪 70 年代早期理想破灭和资金投入大幅削减；从 20 世纪 70 年代第一代专家系统（DENDRAL、MYCIN、PROSPECTOR）的诞生到 20 世纪八九十年代专家系统技术的成熟和其在不同领域的广泛应用；从 20 世纪 40 年代简单的二元神经元模型的提出到 20 世纪 80 年代人工神经网络领域的复苏；从 20 世纪 60 年代模糊集理论的提出和它的存在被西方忽视到 20 世纪 80 年代日本生产出大量模糊用户产品，以及 20 世纪 90 年代软计算和文字计算在世界范围内的广泛接受。

第 2 章提供了基于规则的专家系统概述。作者简要介绍了知识的概念，以及专家用产生式规则表示知识的过程。作者介绍了专家系统开发团队的主要成员和基于规则系统的结构。分析了专家系统的基本特性，并指出专家系统不是万无一失的。然后回顾了前向链接和后向链接推理技术，并讨论了冲突消解策略。最后分析了基于规则的专家系统的优缺点。

第 3 章展示了专家系统使用的两种不确定性管理技术：贝叶斯推理和确信因子。作者分析了不确定知识的主要来源，并简要回顾了概率理论。作者考虑了可累积论据的贝叶斯方法并开发一个简单的基于贝叶斯方法的专家系统。然后讨论确信因子理论（贝叶斯推理的常用替代方法），并开发一个基于论据推理的专家系统。最后，比较贝叶斯推理和确信因子理论并分析它们的适用范围。

第 4 章介绍了模糊逻辑并讨论其背后的哲学思想。首先介绍模糊集的概念，考虑如何在计算机里表示一个模糊集并介绍了模糊集的操作。作者还定义了语言变量和模糊限制语（hedge）。然后作者叙述了模糊规则，并解释了经典规则和模糊规则的主要区别。该章主要研究两种模糊推理技术：Mamdani 法和 Sugeno 法，并就它们适宜的应用领域给出建议。最后介绍开发一个模糊专家系统的主要步骤，并通过构建和调试模糊系统的具体过程来阐明其理论。

第 5 章概述了基于框架的专家系统。介绍了框架的概念，并讨论如何将框架用于知识表达，以及阐明继承是基于框架系统的基本特征，还讨论了方法、守护程序和规则的应用。最后通过一个实例来介绍基于框架的专家系统的开发。

第 6 章介绍了人工神经网络，并讨论了机器学习的基本思想。叙述作为一个简单的计算单元的感知器的概念，并讨论了感知器的学习规则，还探索了多层神经网络，以及讨论了如何提高反向传播学习算法的计算效率。然后介绍循环神经网络，思考 Hopfield 网络训练算法和双向联想记忆（BAM）。最后介绍自组织神经网络并探讨 Hebbian 学习规则和竞争学习。

第 7 章概述了进化计算，其中包括遗传算法、进化策略和遗传编程。首先介绍了开发一个遗传算法的主要步骤，讨论遗传算法的工作机制，并通过具体的遗传算法应用阐明其理论。然后叙述一个进化策略的基本概念，比较了进化策略和遗传算法之间的不同。最后考虑遗传编程以及它的实际应用。

第 8 章讨论了结合不同智能技术的混合智能系统。首先介绍一种新型的专家系统——神经专

家系统，它将神经网络和基于规则的专家系统结合起来。然后考虑一个功能上等同于 Mamdani 模糊推理模型的神经 - 模糊系统，以及一个功能上等同于 Sugeno 模糊推理模型的自适应神经模糊推理系统（ANFIS）。最后讨论进化神经网络和模糊进化系统。

第 9 章讨论了知识工程。首先讨论智能系统可以解决什么样的问题，并介绍知识工程过程的 6 个主要阶段。然后展示了专家系统、模糊系统、神经网络和遗传算法的典型应用。作者演示了如何构建智能系统来解决诊断、选择、预测、分类、聚类和优化问题。最后讨论了混合神经 - 模糊系统在决策支持和时序预测方面的应用。

第 10 章介绍了数据挖掘的概貌，讨论了将数据转换为知识的主要技术。首先，宽泛地定义了数据挖掘，并解释了在大型数据库中进行数据挖掘和知识发现的过程。介绍了一些统计学方法，包括主成分分析，并讨论了它们的局限性。随后展示了关系数据库中结构化查询语言的应用，介绍了数据仓库和多维数据分析。最后，介绍了最流行的数据挖掘工具——决策树和购物篮分析。

本书还包括一个术语表和一个附录。术语表包含了 300 余条用于专家系统、模糊逻辑、神经网络、进化计算、知识工程以及数据挖掘领域的定义。附录中提供了商业化的人工智能工具의列表。

本书的网站地址为 <http://www.booksites.net/negnevitsky>。



在本书的出版过程中，我直接或间接地得到了很多人至诚的帮助。我首先要感谢 Vitaly Fay-bisovich 博士，他不仅对我在软计算领域的研究提出了许多建设性的意见，还在我过去 30 年间的奋斗生涯中与我建立了真诚的友谊并给予了我大力支持。

我还要感谢许许多多为本书提出评论和有益建议的检阅人，感谢 Pearson Education 的编辑们，特别是 Keith Mansfield、Owen Knight、Liz Johnson 和 Rufus Curnow，他们帮助我完成了本书的出版发行工作。

我要感谢我在澳大利亚塔斯马尼亚大学的本科生和研究生们，特别是我以前的博士生 Tan Loc Le、Quang Ha、Steven Carter 和 Mark Lim，他们对新知识的渴求对我来说既是挑战也是激励。

我想感谢美国波士顿大学的 Stephen Grossberg 教授、德国马德堡大学的 Frank Palis 教授、日本广岛大学的 Hiroshi Sasaki 教授、美国罗切斯特理工学院的 Walter Wolf 教授以及东京工业大学的 Kaoru Hirota 教授，他们为我提供了在学生中对书的内容进行测试的机会。

我还要衷心地感谢 Vivienne Mawson 和 Margaret Eldridge 博士对本书初稿的审读工作。

尽管本书的第 1 版出版了十多年，但是已经有很多人用过它并向我提出了宝贵的意见和建议，由于人数众多，在此不能一一致谢，但我至少要感谢那些提出了特别有益建议的人们：Martin Beck（英国普利茅斯大学）、Mike Brooks（澳大利亚阿德莱德大学）、Genard Catalano（美国哥伦比亚大学）、Warren du Plessis（南非比勒陀利亚大学）、Salah Amin Elewa（埃及美洲大学）、Michael Fang（中国浙江大学）、John Fronckowiak（美国曼达尔学院）、Patrick B. Gibson（加拿大温莎大学）、Lev Goldfarb（加拿大 New Brunswick 大学）、Susan Haller（美国威斯康星大学）、Evor Hines（英国沃里克大学）、Philip Hingston（澳大利亚埃迪斯科文大学）、Sam Hui（美国斯坦福大学）、Yong - Hyuk Kim（韩国光云大学）、David Lee（英国赫特福德郡大学）、Andrew Nunekpeku（加纳大学）、Vasile Palade（英国牛津大学）、Leon Reznik（美国罗切斯特理工学院）、Simon Shiu（中国香港理工大学）、Boris Stilman（美国科罗拉多大学）、Thomas Uthmann（德国美因茨约翰尼斯·古腾堡大学）、Anne Venables（澳大利亚维多利亚大学）、Brigitte Verdonk（比利时安特卫普大学）、Ken Vollmar（美国西南密苏里州立大学）、Kok Wai Wong（新加坡南洋理工大学）以及 Georgios N. Yannakakis（丹麦哥本哈根信息技术大学）。

目 录

Artificial Intelligence, 3E

译者序	
第3版前言	
第1版前言	
本书概要	
致谢	
第1章 基于知识的智能系统概述	1
1.1 智能机	1
1.2 人工智能的发展历史, 从“黑暗时代”到基于知识的系统	3
1.2.1 “黑暗时代”, 人工智能的诞生 (1943年~1956年)	3
1.2.2 人工智能的上升期, 远大目标积极实现的年代 (1956年~20世纪60年代晚期)	4
1.2.3 没有履行的诺言, 来自现实的冲击 (20世纪60年代晚期~20世纪70年代早期)	5
1.2.4 专家系统技术, 成功的关键因素 (20世纪70年代早期~20世纪80年代中期)	5
1.2.5 如何使机器学习, 神经网络的重生 (20世纪80年代中期至今)	8
1.2.6 进化计算, 在尝试中学习 (20世纪70年代早期至今)	9
1.2.7 知识工程的新纪元, 文字计算 (20世纪80年代后期至今)	9
1.3 小结	11
复习题	13
参考文献	13
第2章 基于规则的专家系统	16
2.1 知识概述	16
2.2 知识表达技术——规则	16
2.3 专家系统研发团队的主要参与者	18
2.4 基于规则的专家系统的结构	19
2.5 专家系统的基本特征	20
2.6 前向链接和后向链接推理技术	21
2.6.1 前向链接	22
2.6.2 后向链接	23
2.7 MEDIA ADVISOR: 基于规则的专家系统实例	25
2.8 冲突消解	29
2.9 基于规则的专家系统的优点和缺点	31
2.10 小结	32
复习题	33
参考文献	34
第3章 基于规则的专家系统中的不确定性管理	35
3.1 不确定性简介	35
3.2 概率论基本知识	36
3.3 贝叶斯推理	39
3.4 FORECAST: 论据累积的贝叶斯方法	41
3.5 贝叶斯方法的偏差	46
3.6 确信因子理论和基于论据的推理	47
3.7 FORECAST: 确信因子的应用	51
3.8 贝叶斯推理和确信因子的对比	52
3.9 小结	53
复习题	53
参考文献	54
第4章 模糊专家系统	56
4.1 概述	56
4.2 模糊集	57
4.3 语言变量和模糊限制语	60
4.4 模糊集的操作	63
4.5 模糊规则	66
4.6 模糊推理	68
4.6.1 让未来触手可及 Mamdani-style 推理	68
4.6.2 Sugeno-style 推理	73

4.7 建立模糊专家系统·····	75	参考书目·····	169
4.8 小结·····	82	第8章 混合智能系统 ·····	170
复习题·····	83	8.1 概述·····	170
参考文献·····	83	8.2 神经专家系统·····	171
参考书目·····	84	8.3 神经-模糊系统·····	176
第5章 基于框架的专家系统 ·····	86	8.4 ANFIS·····	182
5.1 框架简介·····	86	8.5 进化神经网络·····	188
5.2 知识表达技术——框架·····	87	8.6 模糊进化系统·····	192
5.3 基于框架的系统中的继承·····	91	8.7 小结·····	195
5.4 方法和守护程序·····	94	复习题·····	196
5.5 框架和规则的交互·····	97	参考文献·····	197
5.6 基于框架的专家系统实例: Buy Smart·····	99	第9章 知识工程 ·····	198
5.7 小结·····	108	9.1 知识工程简介·····	198
复习题·····	109	9.1.1 问题评估·····	198
参考文献·····	109	9.1.2 数据和知识获取·····	199
参考书目·····	110	9.1.3 原型系统开发·····	200
第6章 人工神经网络 ·····	111	9.1.4 完整系统开发·····	201
6.1 人脑工作机制简介·····	111	9.1.5 系统评价和修订·····	201
6.2 作为简单计算元素的神经元·····	113	9.1.6 系统集成和维护·····	201
6.3 感知器·····	114	9.2 专家系统可以解决的问题·····	202
6.4 多层神经网络·····	117	9.3 模糊专家系统可以解决的问题·····	209
6.5 多层神经网络的加速学习·····	123	9.4 神经网络可以解决的问题·····	214
6.6 Hopfield 网络·····	126	9.5 遗传算法可以解决的问题·····	226
6.7 双向联想记忆·····	131	9.6 混合智能系统可以解决的问题·····	229
6.8 自组织神经网络·····	133	9.7 小结·····	236
6.8.1 Hebbian 学习·····	133	复习题·····	237
6.8.2 竞争学习·····	136	参考文献·····	239
6.9 小结·····	141	第10章 数据挖掘和知识发现 ·····	241
复习题·····	143	10.1 数据挖掘简介·····	241
参考文献·····	143	10.2 统计方法和数据可视化·····	243
第7章 进化计算 ·····	145	10.3 主成分分析·····	247
7.1 进化是智能的吗·····	145	10.4 关系数据库和数据库查询·····	255
7.2 模拟自然进化·····	145	10.5 数据仓库和多维数据分析·····	258
7.3 遗传算法·····	146	10.6 决策树·····	265
7.4 遗传算法为什么可行·····	153	10.7 关联规则和购物篮分析·····	271
7.5 案例研究: 用遗传算法来维护调度·····	154	10.8 小结·····	277
7.6 进化策略·····	160	复习题·····	278
7.7 遗传编程·····	161	参考文献·····	279
7.8 小结·····	167	术语表 ·····	281
复习题·····	167	附录 人工智能工具和经销商 ·····	295
参考文献·····	168	附录 让未来触手可及 ·····	310

基于知识的智能系统概述

本章讨论何为智能以及机器能否真正实现智能化。

1.1 智能机

哲学家们两千多年来一直致力于理解和解决宇宙的两大问题：人类是如何思考的？人类以外的物体是否也能够思考？至今，这两个问题依旧没有答案。

一部分哲学家接受由计算机领域的科学家提出的计算方法，认可人类所做的任何事情机器都能够做到的观点。另一部分哲学家则非常反对这一观点，认为一些高级复杂的行为（例如爱、创新、道德判断）远超出了任何机器的能力范围。

哲学的特质允许这些争论一直存在。但事实上，工程师和科学家们已经造出了具有“智能”的机器。何为“智能”？下面是来自《柯林斯基础英语词典》的定义：

- (1) 智能是人类理解和学习事情的能力。
- (2) 智能是思考和理解事情的能力，而非本能或机械地做事情。

——《柯林斯基础英语词典》2008 年版

按照第一条定义，智能是专属于人类的一个品质，但第二条定义更具弹性，指出只要具有思考和理解能力就具有智能，而不限主体是人还是物。而定义里的“思考”是什么意思呢？再次看词典中的定义：

思考是使用大脑考虑问题或创建新想法的活动。

——《柯林斯基础英语词典》2008 年版

为了能够思考，人或物需要拥有像大脑那样的一个器官，依赖它去学习、理解事物，去处理问题和做出决策。因此智能可以定义为“学习和理解事物、处理问题并做出决策的能力”。

计算机能否智能化，即机器能否思考，这一问题源于人工智能的“黑暗时代”（20 世纪 40 年代晚期开始）。作为一门科学，人工智能（Artificial Intelligence, AI）的目标是使机器像人那样具有智能去做事情（Boden, 1977）。因此，关于“机器能否思考”这一问题的答案对人工智能这一学科至关重要。答案本身不是“能”或“不能”之类的简单判断，而必定是模糊的。我们从常识中也能得出这个结论：比如一部分人在某些方面比其他聪明；日常生活中做出的抉择有些是明智的，有些则很愚蠢的；有人可能精通复杂的数学、工程问题，但对于哲学或历史却一窍不通；有人善于赚钱，而有人更善于花钱。虽然人类中的每一个成员都具有学习和理解事物、处理问题并做出决策的能力，但比较起来，这种能力在每一个领域都不均等。所以，如果机器能够思考，一部分机器在某些方面也许比另一部分机器聪明。

英国数学家阿兰·图灵（Alan Turing）在 50 多年前撰写的论文“计算机器和智能化”（Turing, 1950）是关于机器智能方面最早和最有影响的文章之一。图灵的思想经住了时间的考验，到现在为止仍是通用的。

阿兰·图灵在 20 世纪 30 年代早期重新发现了中心极限定理，他也由此开始了科学生涯。1937 年，他发表了可计算数字的论文，在这篇文章中提出了通用机器的概念。之后在二战期间，他在破解德国军用编码器 Enigma 的工作中担任了重要角色。第二次世界大战结束后，他设计了“自动计算机”，并设计了第一个国际象棋比赛程序，后来在曼彻斯特大学的计算机上得以实

现。图灵关于通用机的理论概念和译码的实践经验使他在人工智能的关键性基础问题方面取得进展。他曾经提出这样的问题：是否存在不需经验的思考？是否存在不需交流的思考？是否存在不使用的语言？是否存在生命之外的智能？显然，这些问题都是人工智能中的基础问题——“机器能否思考？”的不同表达。

图灵并没有对机器和思考进行定义，而是设计了一个游戏——图灵模拟游戏，用游戏阐述他对机器和思考的理解，从而避免了关于定义的异议。图灵认为，与其问“机器能否思考”，不如问“机器能否通过智能行为测试”。他曾经预测，到公元 2000 年时，计算机能够通过程序与人进行 5 分钟对话，并有 30% 的机会使对话的人认为对方不是计算机，而是人。图灵将计算机智能行为定义为在认知任务中达到人类行为水平的能力。换句话说，如果对话者依据所提问题的答案无法判断对方是人还是机器，计算机就通过了测试。

图灵模拟游戏包括两个阶段。在第一阶段，如图 1.1 所示，一个审讯员和一男一女分别被安排在单独的空间，他们只能通过远程终端之类的中介进行交流。审讯员的目标是通过对外另外两个人提问，根据他们的回答来判断谁是男的、谁是女的。游戏规则是男士要尽力让审讯员认为他是女士，而女士则需向审讯员证明她是女士。

游戏的第二个阶段，如图 1.2 所示，用一台计算机代替男士，计算机的任务和第一阶段中男士的任务一样。对计算机编程时，为了更好地模拟人类，故意让计算机犯错或提供模糊的答案。如果计算机成功“骗过”审讯员的次数和男士成功“骗过”审讯员的次数一致，则认为计算机通过了智能行为测试。

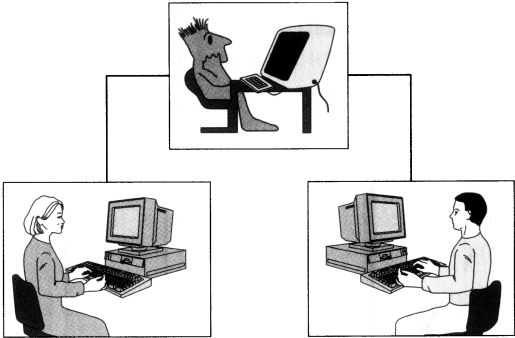


图 1.1 图灵模拟游戏：第一阶段

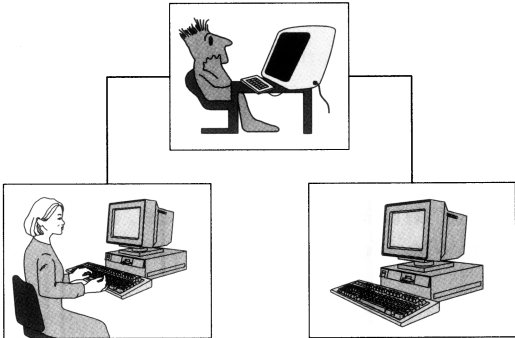


图 1.2 图灵模拟游戏：第二阶段

对于人的身体模拟在智能测试中并不重要。因此，图灵测试中，审讯员看不见、触不到、听不到计算机，因此不会受计算机的外表或声音的干扰。但是为了辨别出计算机，审讯员可以问任何问题，甚至是过激的问题。例如，审讯员可以让人和计算机做复杂的数学计算，如果一方提供了正确答案，并且比对方速度快，可以判断这一方就是计算机。因此，计算机必须知道何时故意出错、何时延迟回答。审讯员也可能会问双方关于短篇小说、诗歌、绘画方面的问题，由此可以推断出一方具有的明显的人类专属的情绪特征。当面对这一类问题时，计算机就必须模拟人类对事物的感性理解。

图灵测试很通用，原因是测试具备了两个显著特征：

- 由于使用终端进行人机交互，这一测试让我们对智能有客观的衡量标准，避免了对人类智能本质的争论，并排除了偏向人类的可能性。
- 测试本身独立于实验细节。既可以像前述的进行两阶段测试，也可以变成一个阶段的测试，从一开始就让审讯员对人与机器进行判断。审讯员既可以提任何领域的任何问题，也可以只关注答案本身。

图灵相信到 20 世纪末，能够通过对数字计算机编程来完成测试游戏。尽管现代计算机还没有通过图灵测试，但已经为我们提供了基于知识的系统的评估和验证基础。通过与人类专家行为对比，已经对一些狭窄专业领域的具有智能的程序进行了评估。

我们的大脑具有 10^{18} 位的容量，能够每秒处理 10^{15} 位的信息。到 2020 年，也许一块方糖大小的芯片就足以模拟人脑，也许会有计算机能够通过图灵测试游戏。但是，我们真的想让计算机在做数学计算时，像人一样又慢又会出错吗？实际上，智能机应该能够帮助人类进行决策、搜索信息、控制复杂对象，最终理解单词含义。也许没有必要片面追求使计算机像人一样聪明这一既抽象又艰涩的目标。为了构建一套智能机系统，我们必须去捕捉、组织和使用某个狭窄专业领域的人类专家知识。

1.2 人工智能的发展历史，从“黑暗时代”到基于知识的系统

人工智能成长为一门科学，历经三代研究者的努力。这一节就来介绍每一代研究者对人工智能学科所做的贡献和重大事件。

1.2.1 “黑暗时代”，人工智能的诞生（1943 年 ~ 1956 年）

Warren McCulloch 和 Walter Pitts 在 1943 年发表了人工智能领域的开篇之作。McCulloch 拥有哥伦比亚大学的哲学和医学学位，后来在伊利诺伊大学的精神病学系担任基础研究实验室的主任。他在对中枢神经系统的研究中提出了大脑神经元模型，最早为人工智能做出了主要贡献。

McCulloch 和年轻的数学家 Walter Pitts 合作，提出了人工神经网络模型，这个模型假定每个神经元有两个状态：开、关（McCulloch and Pitts, 1943）。并证明他们的神经网络模型实际上等价于图灵机，并且任何可计算函数都可以由相连的神经元网络进行计算。同时揭示了简单的网络结构具有学习功能。

神经网络模型从理论和实验两个方面激发了对大脑模型的研究。但是，实验结果清晰地表明将神经元限制为两个状态是不正确的。事实上，神经元具有明显的非线性特征，而非简单的二元状态。即使如此，McCulloch 是人工智能领域继阿兰·图灵之后的第二个创始人，为神经计算和人工智能网络（Artificial Neural Networks, ANN）奠定了基础。人工神经网络的研究在 20 世纪 70 年代经历了衰退，在 20 世纪 80 年代后期重新复苏。

人工智能的第三个奠基人是聪明的匈牙利籍数学家 John von Neumann。他在 1930 年加入了普林斯顿大学，在数学物理系任教。他是阿兰·图灵的同事和朋友。在第二次世界大战期间，von Neumann 在建造原子弹的“曼哈顿项目”中担任重要角色，并成为宾夕法尼亚大学的 ENIAC（Electronic Numerical Integrator and Calculator，电子数字集成计算器）项目的顾问，参与设计了 EDVAC 计算机（Electronic Discrete Variable Automatic Computer，电子离散变量自动计算机）[⊖]。McCulloch 和 Pitts 所提出的神经网络模型使 von Neumann 受到启发。因此，在 1951 年，当来自普林斯顿大学数学系的两个研究生 Marvin Minsky 和 Dean Edmonds 建造第一台神经网络计算机时，von Neumann 鼓励并资助了他们。

在第一代研究者中，不得不提 Claude Shannon。他毕业于麻省理工学院并于 1941 年加入贝尔实验室。Shannon 继承了阿兰·图灵的观点，认为可以使机器智能化。在 1950 年，他发表了一篇关于让机器下象棋的文章，文章指出一盘典型的棋赛大约有 10^{120} 次的移动（Shannon, 1950）。即

⊖ EDVAC 是一个存储程序的机器。

使一台新的 von Neumann 式计算机每微秒就能测试一种走法，移动第一步也得花上 3×10^{106} 年。因此，Shannon 论证了在寻找解决方案时使用启发式策略的必要性。

另一位人工智能奠基人 John McCarthy 也来自普林斯顿大学。他从普林斯顿大学毕业后去达特茅斯学院工作。他说服了 Marvin Minsky 和 Claude Shannon 在达特茅斯学院组织一个暑期研讨会。在 1956 年，他们召集了对机器智能、人工神经网络和自动理论感兴趣的研究者，参加由 IBM 赞助的研讨会。尽管研讨会仅有 10 人参加，但这次会议却促成了人工智能学科的诞生。在之后的 20 年里，参加过达特茅斯学院研讨会的学者及他们的学生在人工智能领域一直起主导作用。

1.2.2 人工智能的上升期，远大目标积极实现的年代（1956 年 ~ 20 世纪 60 年代晚期）

在人工智能学科的发展早期，可以用热情巨大、想法大胆，但成果很少来形容。当时，计算机被用来做常规数学计算才几年时间，而人工智能领域的研究者却已在论证计算机的本领远超过这些。这真是一个有远大目标并积极付诸实现的年代。

John McCarthy 作为达特茅斯学院研讨会的组织者之一，也是“人工智能”这一术语的发明者，从达特茅斯学院转到了麻省理工学院。他设计了高级语言 LISP。LISP 是早期编程语言之一（FORTRAN 仅比它早了两年），到今天还有人在用。在 1958 年，McCarthy 发表了论文“基于常识的程序”，文中提出了一个名为“建议采纳者”（Active Taker）的程序，用于搜索一般性问题的解决方案（McCarthy, 1958）。McCarthy 论证了设计和生成程序的方法，他以驾车去机场为例，采用这个程序基于几个简单的规则设计了驾驶计划。更重要的是，程序无须变动，就可以接受来自不同专业领域的新的规则，即可以接受新知识。因此，“建议采纳者”是第一个真正集成了知识表达和推理的核心法则的基于知识的系统。

达特茅斯学院研讨会的另一个组织者 Marvin Minsky 也加入了麻省理工学院。他不像 McCarthy 那样专注于形式逻辑，而是建立了一套反逻辑的观点，以处理知识表达和推理。他的框架理论为知识工程做出了主要的贡献（Minsky, 1975）。

由 McCulloch 和 Pitts 开创的神经计算和人工神经网络的早期研究得以继续。随着学习方法的改进和提高，Frank Rosenblatt 证明了感知器收敛理论，并论证他的学习算法能够调整感知器的连接程度（Rosenblatt, 1962）。

GPS（General Problem Solver，通用问题解决方案）是在这个大胆假设年代最具野心的项目之一（Newell and Simon, 1961；1972）。卡内基 - 梅隆大学的 Allen Newell 和 Herbert Simon 设计了一个通用程序，用于模拟人类在解决问题时的方法。GPS 也许是第一个尝试将解决问题的技术和数据分开的程序。它使用了现在已被称为手段-目的分析（means-ends analysis）的技术。Newell 和 Simon 假定可以用状态去定义问题。手段 - 目的的作用是判断问题的当前状态和期望状态或目标状态的差距，并选择操作算子去达到目标状态。如果从当前状态不能直接到达目标状态，就要建立一个更接近目标状态的新状态，重复这一过程直到达到目标状态。操作算子集合决定了解决方案。

但是，GPS 无法解决复杂问题。由于程序建立在形式逻辑的基础上，操作算子可能有无穷多个，导致效率低下。在解决实际问题时，GPS 需要大量运算时间和内存，这导致人们后来放弃 GPS 计划。

总之，在 20 世纪 60 年代，为了模拟复杂的思考过程，人工智能研究者试图研究通用方法来解决广泛的问题。他们使用了通用搜索策略以寻找解决方案。通用搜索策略现在被称为弱方法，由于使用的是问题域的弱信息，导致程序的性能低下。

不管怎样，这个发展阶段吸引了伟大的科学家针对人工智能各个方面提出创新性的基础理论，例如在知识表达、学习算法、神经计算以及文字计算等诸方面。当时由于计算机性能的限制，这些理论并未实现，但却为20年后的实际应用指明了方向。

值得一提的是，来自加州大学伯克利分校的 Lotfi Zadeh 教授也在20世纪60年代发表了著名的文章“模糊集”（Zadeh, 1965）。现在人们一致认为这篇文章为模糊集理论奠定了基础。20年后，模糊集理论研究者已经构建了成百上千的智能机和智能系统。

人工智能令人鼓舞的时代到1970年结束，大部分政府资助的人工智能项目也被迫取消。人工智能仍是一个相对新的领域，偏于学术性，除了一些游戏之外几乎没有实际应用（Samuel, 1959; 1967; Greenblatt et al., 1967）。在外界看来，由于不存在能解决实际问题的人工智能系统，所取得的成绩也只是小玩意而已。

1.2.3 没有履行的诺言，来自现实的冲击（20世纪60年代晚期~20世纪70年代早期）

从20世纪50年代中期，人工智能研究者就承诺到20世纪80年代建造出与人类智能相当的全能智能机，到公元2000年，建造出超越人类智能的智能机。但到1970年时，他们已经意识到这些许诺过于乐观。虽然个别智能程序能够在一两个小问题上展示出一定水平的智能，但几乎所有的人工智能项目都对稍宽泛的问题和实际的复杂问题无能为力。

在20世纪60年代晚期，人工智能学科存在的问题主要表现在以下几个方面。

- 由于人工智能研究者专注于开发能解决广义问题的一般方法，因此早期的程序只包含很少甚至没有相关问题域的知识。为了解决问题，程序中应用了一种搜索策略，尝试对小步骤进行不同的组合，直到找出真正的解决方案。由于这种思路对于小问题是有效的，因而被误认为可以通过对程序进行扩展来解决更大的问题。事实上，这种想法是错误的。

简单的而易处理的问题在多项式级的时间内就得以解决，就是说，对于一个规模为 n 的问题，所需的求解时间或求解步骤数是 n 的多项式函数。而解决棘手而不易处理的问题所需的时间是问题规模的指数级函数。一般认为，多项式级运算时间的算法是高效的，而指数级运算时间的算法则是低效的，原因是后者的运算时间随着问题规模的增大而呈飞速增长。20世纪70年代早期所提出的 NP 完全问题理论（NP-Completeness）（Cook, 1971; Karp, 1972）揭示出一大类非确定多项式级问题（Non-deterministic Polynomial problem, NP 问题）都是 NP 完全问题。NP 问题是指能够在多项式时间内估算并进行验证的问题（假设有解）。非确定是指不存在特定算法来进行估算。而 NP 完全问题则是 NP 问题中最难的问题，即使用速度最快、内存最大的计算机也难以求解。

- 许多人工智能试图解决的问题都太宽泛、太复杂。早期人工智能的一个典型任务就是机器翻译。例如，在前苏联于1957年发射第一颗人造卫星 Sputnik 之后，美国国家研究委员会就资助针对俄语科学文献的机器翻译项目。最初，项目团队想利用电子词典直接将俄语单词替换为对应的英语单词。不久他们便发现只有了解了文献主题才能正确选择单词。由于任务太难，1966年由美国政府资助的所有翻译项目都被取消。
- 在1971年，英国政府也终止了对人工智能研究的资助。英国科学研究委员会派 James Lighthill 爵士调研当时的人工智能研究现状，他发现人工智能领域并没有重大甚至显眼的研究成果，因而认为不必保留一个独立的“人工智能”学科。

1.2.4 专家系统技术，成功的关键因素（20世纪70年代早期~20世纪80年代中期）

20世纪70年代最重要的进展，也许就是人们意识到必须对智能机器的问题域进行充分限制。而在此之前，人工智能研究者一直认为，为了对通用的、人类惯用的解决问题的方法进行仿

真，必须创造出聪明的搜索算法和推理技术。一种通用的搜索机制可以依赖基本的推理步骤来发现完整的解决方案并可以使用问题域的弱知识。然而，当这种弱方法失败后，研究者们最终意识到，唯一出路是使用大量推理步骤来解决狭窄专业领域的典型问题。

DENDRAL 是当时的一个典型例子 (Buchanan et al., 1969)，其由斯坦福大学开发，用于化学分析。NASA 资助了这个项目，原因是当时美国要发射一个无人宇宙飞船到火星，需要根据由质谱仪提供的大量光谱数据，由此设计程序以分析火星表面土壤的分子结构。Edward Feigenbaum (曾是 Herbert Simon 的学生)、Bruce Buchanan (计算机科学家)、Joshua Lederberg (遗传学领域的诺贝尔奖获得者) 组成团队来解决这一挑战性的问题。

这类问题的传统解决方法都依赖于枚举—测试技术：首先枚举出与质谱图一致的所有可能的分子结构，用每一个分子结构预测质谱，再根据实际的质谱进行检验。但是，由于分子结构会达到千百万个，即使分子数目适中，问题也会迅速复杂化，因此这种方法是行不通的。

更困难的是，没有科学的算法将质谱图匹配到对应的分子结构。不过，像 Lederberg 这样的分析化学家可以使用他们的技巧、经验和特长解决这个问题。通过找出质谱图中的高峰期所具有的明显的模式，可以大大降低可能的分子结构数目，再使用几个可行的方法进一步求解。所以，Feigenbaum 的任务就是在程序设计时将 Lederberg 的专业知识考虑进来，以使程序具有相当于人类专家的水平。这种程序后来被称为专家系统。要想理解、采纳 Lederberg 的知识，并用他的术语进行操作，Feigenbaum 需要学习化学和质谱分析的基本知识。当然，他不仅需要使用化学规则，也需要基于其经验和猜测，并使用他自己的经验法则之类的启发式方法。不久 Feigenbaum 就发现项目中的一个主要困难是如何从人类专家那里抽取知识并应用于计算机，他将这一问题称为“知识获取瓶颈”。为了更清楚地表达知识，Lederberg 也需要学习计算方面的基础知识。

Feigenbaum、Buchanan 和 Lederberg 组成团队，最终成功开发出了第一个基于知识的系统——DENDRAL。他们成功的关键在于将所有相关的理论知识从泛化形式映射到极为具体的规则(“烹饪食谱”) (Feigenbaum et al., 1971)。

DENDRAL 的巨大意义可以归纳为如下几点：

- DENDRAL 标志着人工智能领域重要的“范式转移”，即从通用的、知识稀疏型弱方法转移到针对特定领域的知识密集型技术。
- DENDRAL 项目的目标是设计出能达到有经验的化学家水平的计算机程序。通过提取人类专家知识，并表达为高质量的具体规则（经验法则），DENDRAL 团队使用这种启发式策略证明了计算机在特定、可定义的问题领域能达到专家水平。
- DENDRAL 为专家系统开创了新的方法论，即知识工程，它包含了对专家知识从获取、分析到用规则表达等一系列技术。

DENDRAL 后来成为化学家们有用的分析工具，并在美国成为商业产品被投放到市场。

Feigenbaum 和来自斯坦福大学的人员接下来承担了医疗诊断方面的一个重大项目，这个项目名为 MYCIN，始于 1972 年，后来成为 Edward Shortliffe 的博士论题 (Shortliffe, 1976)。MYCIN 是基于规则的专家系统，它的任务是诊断传染性的血液病，它还以一种便捷、用户友好的方式为医生提供治疗建议。

MYCIN 与早期专家系统具有若干相同特点：

- MYCIN 的性能相当于人类专家水平，并高于初级医生的水平。
- MYCIN 包含了 450 条独立的 IF-THEN 形式的规则，这些知识是通过访问特定领域的大量专家获得的。
- 以规则形式表达的知识与推理机制明确地分开了。系统研发员也能够轻易地通过插入、删除规则来操纵知识。例如，斯坦福大学后来研发了一个独立于领域知识的 MYCIN 版本，

称为 EMYCIN (van Melle, 1979; van Melle et al., 1981)。它除了传染性血液病知识之外, 具有 MYCIN 的一切特征。EMYCIN 促进了诊断方面的各种应用。系统研发员只需将新知识以规则形式添加, 就能实现新的应用。

与早期专家系统相比, MYCIN 也有一些新特点。例如, MYCIN 的规则反映了知识本身的不确定性, 这里是指医学诊断知识。它在可利用的数据或医生所给的数据上测试规则的条件部分 (IF 部分)。在适当时候, MYCIN 使用名为确信心因子的不确定性运算推断是否满足条件。考虑不确定性的推理是这个系统的最重要的特征。

另一个被广泛关注的概率系统是 PROSPECTOR, 这一专家系统由斯坦福研究院开发, 用于矿产勘探 (Duda et al., 1979)。这个项目始于 1974 年, 在 1983 年结束, 有 9 位专家为该系统提供了专业知识。PROSPECTOR 将规则和语义网络合并到一个结构中, 以表达专家知识, 并使用了一千多条规则来表达大量领域知识。同时拥有一个复杂的支持包, 其中包括知识获取系统。

PROSPECTOR 的工作原理是, 首先让作为用户方的勘探地质学家输入待检矿床的特征, 如地址环境、结构、矿物质类型等。之后, 程序将这些特征与矿床模型比较, 必要时让用户提供更多信息。最后, 系统对待检矿床做出结论。这一系统也能够解释为了得出结论所用到的步骤。

在勘探地质学领域, 重要决策常常是在由于知识不完整或模糊而导致不确定性的情况下做出的。为了处理这类知识, PROSPECTOR 使用论据贝叶斯规则在系统中传送不确定性。它的性能达到了专业地质学家的水平, 并且在实践中得到了验证。在 1980 年, 用它识别出了华盛顿州 Tolman 山附近的一个钼矿床。随后一个采矿公司对这个矿床开采时证实这个矿床价值一亿美元。真是难以想象专家系统能有如此价值。

上面所提到的几个专家系统现在已成为经典之作。20 世纪 70 年代后期, 专家系统越来越多地被成功应用, 这表明能够将人工智能技术从实验室成功地移植到商业领域。不过, 这一时期大部分专家系统都是在功能强大的工作站上, 采用特殊的人工智能语言开发的, 例如 LISP、PROLOG 以及 OPS。昂贵的硬件和复杂的程序语言意味着专家系统的开发工作只能由个别院校的研究组做, 像斯坦福大学、麻省理工学院、斯坦福研究院和卡内基 - 梅隆大学。到了 20 世纪 80 年代, 个人电脑和易用的专家系统开发工具框架的出现, 才使得普通的研究者和各个领域的工程师都有机会开发专家系统。

1986 年的一份综述报告汇总了大量成功应用于不同领域的专家系统, 所应用的领域包括化学、电子、工程、地质、管理、医药、过程控制和军事科学, 等等 (Waterman, 1986)。尽管如此, 在 Waterman 发现的将近 200 个专家系统中, 大部分还是应用在医学诊断领域。7 年后另一篇综述汇总了 2500 多个开发完毕的专家系统 (Durkin, 1994)。新兴的应用领域是商业和制造业, 占了总应用的 60%。此时专家系统技术已经成熟。

专家系统在任何领域都是成功的关键吗? 虽然有了大量的专家系统, 并且在不同知识领域得以成功地应用, 但高估这一技术的能力是不对的。在技术和社会学两个层面都存在复杂的困难, 这些困难包括:

- 专家系统局限于非常狭窄的专业领域。例如, 虽然 MYCIN 的任务是诊断传染性血液病, 却没有人体生理学的知识。如果病人的病不止一个, 就不能指望 MYCIN 了。事实上, 当病人伴随其他疾病时, 使用 MYCIN 开的治疗血液病的处方甚至可能是有害的。
- 由于局限于狭窄的领域, 专家系统并不能如用户所愿的那样健壮灵活。并且, 专家系统难以识别领域界限。当一项任务不同于传统问题时, 专家系统可能在尝试解决时出乎意料地失败。
- 专家系统的解释能力有限。虽然能够显示解决方案中应用的一系列规则, 却无法将累积的启发式知识与对问题领域的深层理解关联起来。

- 专家系统难以检验验证。至今都没有开发出通用的技术来分析专家系统的完整性和一致性。启发式规则以抽象形式表达知识，缺乏对领域的基本理解，这使得识别错误的、不完整或不一致的知识的工作非常困难。
- 专家系统几乎不具备从它们的经验中去学习的能力，尤其是第一代专家系统。专家系统的开发相对独立而且开发过程慢。解决一个中等难度的问题要花费 5 ~ 10 人年的时间 (Waterman, 1986)。而像 DENDRAL、MYCIN 或 PROSPECTOR 之类的复杂系统需要超过 30 人年的开发时间。如果专家系统的完善依赖于它的开发者进一步关注，那么这些巨大的投入将难以评估。

尽管面临这么多的困难，专家系统的研究还是有了很大突破，并在若干重要应用中证明了它的价值。

1.2.5 如何使机器学习，神经网络的重生 (20 世纪 80 年代中期至今)

在 20 世纪 80 年代中期，研究者、工程师以及专家们意识到，购买一个推理系统或专家系统框架，再往里填充足够的规则，并不足以构造一个专家系统。应用专家系统技术的幻想破灭，对人工智能项目的资助也严重紧缩，甚至导致人们开始预测人工智能的“冬季”将要到来。人工智能研究者们决定重新审视神经网络。

到 20 世纪 60 年代后期，神经计算所需的基本理论和概念大部分都已经成型 (Cowan, 1990)。但是，直到 20 世纪 80 年代中期才出现解决方案，滞后的主要原因在于技术层面：没有个人电脑或功能强大的工作站，所以无法对人工神经网络建模和实验。还有心理学和资金方面的原因。例如，在 1969 年，Minsky 和 Papert 就用数学方法论证了单层感知器具有根本的计算局限性 (Minsky and Papert, 1969)，他们还认为即使复杂的多层感知器也不见得更好。这显然不会激励人去研究感知器，这种情况导致在 20 世纪 70 年代几乎所有的人工智能研究者都放弃了人工神经网络领域。

在 20 世纪 80 年代，由于需要像大脑那样进行信息处理，以及计算机技术的发展和神经科学的进步，神经网络领域快速复苏了。在几个前沿领域，科学家们在理论和设计上都做了重大贡献。Grossberg 创建了自组织的新理论 (adaptive resonance theory, 自适应共振理论)，这为一种新的神经网络奠定了基础 (Grossberg, 1980)。Hopfield 引入了具有反馈机制的神经网络，即 Hopfield 网络 (Hopfield networks, Hopfield, 1982)，在 20 世纪 80 年代引起了广泛关注。Kohonen 发表了一篇关于自组织映射的文章 (Kohonen, 1982)。Barto、Sutton 和 Anderson 发表了关于增强学习 (reinforcement learning) 和在控制领域的应用的文章 (Barto et al., 1983)。但真正的突破是在 1986 年，Rumelhart 和 McClelland 在《并行分布式处理：对认知微结构的探索》一文中，重新提出了反向传播学习算法 (back-propagation learning algorithm) (Rumelhart and McClelland, 1986)，这一算法由 Bryson 和 Ho 在 1969 年首次提出 (Bryson and Ho, 1969)。Parker (Parker, 1987) 和 LeCun (LeCun, 1988) 在同一时间也提出了反向传播学习算法，从此这一算法成为用于训练多层感知器的最流行的技术。在 1988 年，Broomhead 和 Lowe 提出了使用径向基函数设计多层前馈网络的程序，这是多层感知器的替代 (Broomhead and Lowe, 1988)。

从 McCulloch 和 Pitts 提出的早期模型，发展为根植于神经科学、心理学、数学和工程的交叉学科，人工神经网络经过了一个漫长的历程，并在理论和应用两方面还会继续发展。无论如何，对于 20 世纪 80 年代神经网络的重生，Hopfield 的文章 (Hopfield, 1982)、Rumelhart 和 McClelland 的书 (Rumelhart and McClelland, 1986) 具有最重要、最具影响的意义。

1.2.6 进化计算，在尝试中学习（20 世纪 70 年代早期至今）

自然界的智能是进化的产物。因此，通过模拟生物进化，我们有希望找出将现有的系统推向高水平智能的方法。自然界是在尝试中学习的，并没有外界告诉生物系统如何适应具体环境，这是物竞天择的过程。适应性最强的物种更有机会繁衍，并将基因遗传给下一代。

人工智能中的进化方法是基于自然选择和遗传的计算模型。进化计算的工作流程包括模拟由个体组成的种群、评估个体性能、产生下一代种群，这一过程需要迭代若干次。

进化计算主要包括 3 个主要技术：遗传算法、进化策略以及遗传编程。

遗传算法是 John Holland 在 20 世纪 70 年代早期提出的（Holland, 1975）。他设计了一个包括选择、交叉、变异 3 个遗传操作的算法，用于操纵人工“染色体”（表达为二进制字符串）。模式定理是遗传算法的坚实的理论基础（Holland, 1975；Goldberg, 1989）。

在 Holland 的遗传算法之前，早在 20 世纪 60 年代早期，两个来自柏林科技大学的学生 Ingo Rechenberg 和 Hans-Paul Schwefel 提出了一个新的名为进化策略（evolutionary strategies）的优化方法（Rechenberg, 1965）。进化策略的具体任务是解决工程设计中的参数优化问题。Rechenberg 和 Schwefel 建议，像自然界变异那样对参数随机变化。事实上，可将进化策略方法视为工程师直觉的替代物。进化策略使用一种数值优化过程，类似于蒙特·卡洛搜索方法。

遗传算法和进化策略都可以解决广泛的问题。这两个方法为之前根本无法解决的高度复杂的非线性搜索和优化问题提供了健壮、可靠的解决方案（Holland, 1995；Schwefel, 1995）。

遗传编程代表了将学习式的遗传模型在程序设计中的应用，它的目的并不是推出问题的编码表示，而是推出用以解决问题的计算机代码。就是说，遗传编程的目的是产生解决问题的计算机程序。

20 世纪 90 年代 John Koza 的工作极大促进了人们对遗传编程的兴趣（Koza, 1992；1994），他使用遗传算子来操纵表示 LISP 程序的符号码。遗传编程为计算机科学领域的主要问题提供了解决方案，使计算机不需通过精确编程就能解决问题。

遗传算法、进化策略和遗传编程标志着人工智能学科的快速成长，并且具有巨大的发展空间。

1.2.7 知识工程的新纪元，文字计算（20 世纪 80 年代后期至今）

与基于符号推理的系统相比，神经网络的技术提供了与现实世界更加自然的交互。神经网络能够学习、适应问题环境的变化，能在规则未知的情形下构建模式，并能处理模糊、不完整的信息。但是神经网络的解释功能差，像黑匣子一样工作。用现有的技术来训练神经网络很耗时，而且频繁的重新训练还会导致严重的困难。

尽管在一些特殊的环境，尤其是缺少知识的环境中，人工神经网络能比专家系统更好地解决问题，但这两个技术发展到现在已不再彼此竞争，而是互补。

传统的专家系统很适用于具有精确输入和逻辑输出的封闭式系统的应用。这类系统使用表达为规则形式的专家知识，必要时还可以与用户交互以构建特定的事实。这类系统的主要缺点在于人类专家并不总能用规则表达知识或逐条解释推理，从而会妨碍专家系统积累必需的知识，最终导致失败。为了克服这个缺陷，可使用神经计算从大数据集中抽取隐含的知识以获取专家系统的规则（Zahedi, 1993；Medsker and Leibowitz, 1994）。在传统的基于规则的专家系统中也可以使用人工神经网络来校正规则（Omlin and Giles, 1996）。换句话说，当得到的知识不完整时，可以用神经网络精炼知识，当知识与给定的数据不一致时，可使用神经网络修改规则。

另一个非常重要的技术是模糊逻辑，用于处理模糊、不精确、不确定的知识和数据。传统的专家系统中，用于处理不精确数据的大部分方法都是基于概率论思想，如 MYCIN 就引入了确信因子，PROSPECTOR 则使用了贝叶斯规则来传送不确定性。不过，专家们并不总是使用概率值，

而会使用“常常、一般地、有时、偶尔、极少”之类的术语。模糊逻辑正是关于模糊值使用的方法，模糊值用于理解单词含义、人类推理、制定决策。模糊逻辑以能精确反映专家对复杂难题的理解的形式，对人类知识进行编码和应用，从而为打破传统专家系统的计算瓶颈提供了一个可行的方法。

模糊逻辑的核心在于语言变量这一概念。语言变量的值是文字，而不是数值。模糊系统使用 IF-THEN 规则来体现人类知识，这点与专家系统类似，只是模糊系统的规则是模糊的，例如：

IF speed is high THEN stopping_distance is long

IF speed is low THEN stopping_distance is short

模糊逻辑或模糊集理论是由加州大学伯克利分校电子工程系主任 Lotfi Zadeh 教授在 1965 年提出的 (Zadeh, 1965)，它提供了一种采用文字进行计算的方法。然而，模糊集理论被技术团体接受的过程却很困难和漫长，部分原因在于具有争议的名字“模糊”，这个名字似乎太随意而难以被认真对待。模糊理论虽然在西方被忽视，却在东方的日本被接纳了。自 1987 年起，模糊理论被成功地应用在日本人设计的洗碗机、洗衣机、空调、电视机、复印机和汽车上。

模糊产品的出现使得这一看似新颖而在 30 年前就被提出的技术受到极大关注，成百上千的书籍和学术论文相继诞生。一些经典著述包括：《Fuzzy Sets, Neural Networks and Soft Computing》(Yager and Zadeh, eds, 1994)；《The Fuzzy Systems Handbook》(Cox, 1999)；《Fuzzy Engineering》(Kosko, 1997)；《Expert Systems and Fuzzy Systems》(Negoita, 1985)；以及畅销书《Fuzzy Thinking》(Kosko, 1993)，这本书对模糊逻辑进行了普及性介绍。

模糊逻辑主要应用在控制工程领域。不过，模糊控制系统仅使用了模糊逻辑知识表达的一小部分功能。模糊逻辑在基于知识的系统和决策支持系统中的应用优势可归纳如下 (Cox, 1999; Pedrycz and Gomide, 2007; Turban et al., 2010)：

- 提高了计算能力。基于规则的模糊系统比传统专家系统运算快，需要的规则也少。模糊专家系统通过合并规则使系统功能更强。Lotfi Zadeh 认为用不了几年，多数专家系统都将使用模糊逻辑来解决高度非线性问题和计算困难的问题。
- 改善了认知模型。模糊系统允许以能反映专家思考复杂问题的形式对知识编码。专家在思考复杂问题时，会用到不精确术语，例如高和低、快和慢、轻和重，也用使用术语常常和几乎不、通常和很少、频繁和偶尔。在建立传统的规则时，需要对术语定义严格的界限，这就把专业知识分解成了知识片段。当传统的专家系统面对高度复杂问题时，知识片段将导致系统性能低下。相反地，模糊专家系统通过对不精确信息建模，就能以更接近专家思考的方式捕捉专业知识，从而改善了系统的认知建模。
- 能够表达多个专家。传统的专家系统建立在非常狭窄的、被明确定义的专业知识领域，系统性能完全依赖于能否正确选择专家。尽管一般的策略是只选定一个专家，但是当构造更复杂的专家系统或专业知识不能被较好地定义时，就需要多个专家。多个专家能够扩展领域，综合经验，并且避免了对顶级专家的依赖，顶级专家一般费用昂贵且难以联系到。不过，多个专家极难达成一致意见，他们的意见常常不同甚至有冲突。在经济和管理领域，当没有简单方案而又需考虑各种具有冲突的观点时尤其如此。模糊专家系统有助于表达意见相左的多个专家的专业知识。

尽管模糊系统允许以更自然的方式表达专家知识，但是它们仍依赖来自专家提炼的规则，因而系统有聪明的，也有笨拙的。有些专家能够提供非常聪明的模糊规则，有些专家只会猜测或提供错误的规则。所以，所有的规则必须经过测试、调整，这一过程可能会漫长又乏味。例如，日立公司的工程师们用了好几年时间才测试、调整了仙台地铁系统上的 54 条模糊规则。

借助模糊逻辑开发工具，能够轻松地构造出一个简单的模糊系统，但之后很可能要花费几

天、几个星期甚至几个月来测试新规则并调整系统。怎样加快这个过程呢？或者，怎样自动产生好的模糊规则呢？

近年来，一些基于神经网络技术的方法已被用于在数值数据中寻找模糊规则。自适应或神经-模糊系统能够发现新的模糊规则，或者基于提供的数据来调整已有的规则。换言之，即是输入数据-输出规则，或者输入经验-输出规律。

那么知识工程将向什么方向发展呢？

专家系统、神经网络和模糊系统目前已经成熟，并被广泛应用于多个领域处理不同问题，主要在工程、医药、金融、商业和管理等领域。各个技术在处理人类知识的不确定性和模糊性上都各有方法，在知识工程领域都占有一席之地。这 3 个技术之间不再竞争，而是互相配合。在专家系统中融入模糊逻辑和神经计算，能够提高基于知识系统的自适应性、鲁棒性、容错能力和运算速度。此外，利用文字进行计算使得系统更加“人性化”。在构造智能系统时，使用现有理论而非提出新理论已成为惯例。应用目标也不再限于简单的小问题，而是实在的现实问题。

1.3 小结

我们处在知识革新的时代，在这个时代，一个国家的实力不是由军队的士兵数量来决定，而是由这个国家所拥有的知识来决定。科学、医药、工程和商业一方面推动着国家朝更高质量的生活发展，一方面也需要高素质的人。当前我们已开始使用智能机器从知识渊博的人身上获取经验知识，并让机器像人那样进行推理。

在第一台计算机发明之前，对智能机器的渴求仅仅是一个渺茫的梦。虽然早期的计算机按照特定的算法能够有效操作大型数据库，却不能对提供的信息进行推理，因而引发了计算机能否思考的问题。阿兰·图灵将计算机的智能行为定义为在认知任务中达到人类同等水平的能力。图灵测试为基于知识的系统提供了检验和验证的基础。

1956 年，在达特茅斯学院召开的暑期研讨会将 10 个对机器智能感兴趣的研究者聚到了一起，一门新学科——人工智能诞生了。

自从 20 世纪 50 年代早期，人工智能技术从个别研究者的好奇开始，发展为帮助人类决策的有价值的工具。从 20 世纪 60 年代关于人工智能的伟大幻想和期待，到 20 世纪 70 年代早期对人工智能的幻灭和资助削减；从 20 世纪 70 年代的第一代专家系统的开发，如 DENDRAL、MYCIN 和 PROSPECTOR，到 20 世纪八九十年代专家系统的成熟以及在不同领域的广泛应用；从 20 世纪 40 年代所提的简单的二进制神经元模型，到 20 世纪 80 年代人工神经网络的戏剧性复活；从 20 世纪 60 年代所提出的模糊集理论在西方被忽视，到 20 世纪 80 年代由日本提供的大量的“模糊型”消费品，以及 20 世纪 90 年代软计算和用文字计算被全世界接受；这些事件使我们看到了人工智能的历史性发展周期。

专家系统的发展创立了知识工程，它是一种构建智能系统的过程。如今知识工程不仅涉及专家系统，还涉及神经网络和模糊逻辑。知识工程仍然是一门艺术，而非工程，但人们已经尝试使用神经网络技术从数值数据中自动抽取规则。

表 1.1 总结了人工智能和知识工程发展历程中的关键事件，涵盖范围从 1943 年 McCulloch 和 Pitts 在人工智能上的初次尝试，到近年来的用文字计算的基于知识的系统中将专家系统、模糊逻辑和神经计算进行合并，从而各取所长。

本章的主要内容包括：

- 智能是指学习和理解问题、解决问题、制定决策的能力。
- 人工智能是使机器像人那样用智能做事的一门学科。
- 如果机器在某些认知任务中能达到人类同等水平，它就是智能机器。在建造智能机器时，

我们必须捕捉、组织和使用特定问题领域的人类专家知识。

- 必须严格限制智能机器的问题域，对此的确认标志着人工智能从泛化的、知识稀疏的弱方法向具体领域的知识密集型方法的重大“范式转移”。这一转移促成了专家系统的发展，从而使计算机程序的性能在狭窄问题领域达到人类专家水平。专家系统以特定规则的形式使用人类知识和经验，主要特征表现在对知识和推理机制的明确划分。专家系统更能解释推理过程。

表 1.1 人工智能和知识工程发展历程中主要事件一览表

时 期	主 要 事 件
人工智能的诞生 (1943 年 ~1956 年)	McCulloch and Pitts, <i>A Logical Calculus of the Ideas Immanent in Nervous Activity</i> , 1943 Turing, <i>Computing Machinery and Intelligence</i> , 1950 电子数据集成器和计算器项目 (von Neumann) Shannon, <i>Programming a Computer for Playing Chess</i> , 1950 在达特茅斯学院举行的机器智能、人工智能和自动化理论暑期研讨会, 1956
人工智能的崛起 (1956 年 ~20 世纪 60 年代晚期)	LISP (McCarthy) 通用问题解决方案 (GPR) 项目 (Newell and Simon) Newell and Simon, <i>Human Problem Solving</i> , 1972 Minsky, <i>A Framework for Representing Knowledge</i> , 1975
人工智能的幻灭 (20 世纪 60 年代晚期 ~ 20 世纪 70 年代早期)	Cook, <i>The Complexity of Theorem Proving Procedures</i> , 1971 Karp, <i>Reducibility Among Combinatorial Problems</i> , 1972 Lighthill 报告, 1971
专家系统的发明 (20 世纪 70 年代早期 ~ 20 世纪 80 年代中期)	DENDRAL (Feigenbaum, Buchanan and Lederberg, 斯坦福大学) MYCIN (Feigenbaum and Shortliffe, 斯坦福大学) PROSPECTOR (斯坦福大学研究院) PROLOG ——一种逻辑编程语言 (Colmerauer, Roussel and Kowalski, France) EMYCIN (斯坦福大学) Waterman, <i>A Guide to Expert Systems</i> , 1986
人工智能的重生 (20 世纪 80 年代 中期至今)	Hopfield, <i>Neural Networks and Physical Systems with Emergent Collective Computational Abilities</i> , 1982 Kohonen, <i>Self-Organized Formation of Topologically Correct Feature Maps</i> , 1982 Rumelhart 和 McClelland, <i>Parallel Distributed Processing</i> , 1986 首届 IEEE 神经网络国际会议 Haykin, <i>Neural Networks</i> , 1994 神经网络, MATLAB 应用工具箱 (The MathWorks, Inc.)
进化计算 (20 世纪 70 年代早期至今)	Rechenberg, <i>Evolutionsstrategien - Optimierung Technischer Systeme Nach Prinzipien der Biologischen Information</i> , 1973 Holland, <i>Adaptation in Natural and Artificial Systems</i> , 1975 Koza, <i>Genetic Programming: On the Programming of the Computers by Means of Natural Selection</i> , 1992 Schwefel, <i>Evolution and Optimum Seeking</i> , 1995 Fogel, <i>Evolutionary Computation-Towards a New Philosophy of Machine Intelligence</i> , 1995
用文字计算 (20 世纪 80 年代 晚期至今)	Zadeh, <i>Fuzzy Sets</i> , 1965 Zadeh, <i>Fuzzy Algorithms</i> , 1969 Mamdani, <i>Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis</i> , 1977 Sugeno, <i>Fuzzy Theory</i> , 1983 日本的“模糊型”消费品 (洗碗机、洗衣机、空调、电视机、复印机) 仙台地铁系统 (日立公司, 日本), 1986 Negoita, <i>Expert Systems and Fuzzy Systems</i> , 1985 首届 IEEE 神经网络国际会议, 1992, Kosko, <i>Fuzzy Thinking</i> , 1993 Yager and Zadeh, <i>Fuzzy Sets, Neural Networks and Soft Computing</i> , 1994 Cox, <i>The Fuzzy Systems Handbook</i> , 1994 Zadeh, <i>Computing with Words-A Paradigm Shift</i> , 1996 模糊逻辑, MATLAB 工具箱 (The MathWorks, Inc.) 神经网络, MATLAB 工具箱 (The MathWorks, Inc.) 伯克利软计算倡议组织 (BISC) http://www-bisc.cs.berkeley.edu

- 构建智能机器面临的主要问题是知识获取瓶颈，即从人类专家那里获取知识，这也是知识工程的主要问题。
- 专家在思考时会使用不精确的术语，如常常和几乎不、通常和很少、频繁和偶尔，也会使用语言变量，如高和低、快和慢、轻和重。模糊逻辑或模糊集理论为用文字进行计算提供了方法。模糊逻辑的重心是使用模糊值，以捕捉和理解文字含义、人类的推理和决策的制定，并为传统专家系统在计算上的负担问题提供了突破性的方法。
- 专家系统既不能通过经验学习、也不能通过经验自我完善，多由个人开发，后续发展也需要巨大的投入，即使构建中型专家系统也需要 5~10 年。机器学习则大大加快了专家系统的开发进程，通过加入新规则或校正已有规则，也提高了知识的质量。
- 人工神经网络启发于生物学上的神经网络，基于历史事件进行学习，并能自动生成规则，由此避免了冗长、昂贵的知识获取过程，以及相应的验证、调整过程。
- 通过对专家系统、人工神经网络分别与模糊逻辑的集成，提高了基于知识的系统的自适应性、容错能力和运算速度。

复习题

- 1.1 何为智能？什么是机器的智能行为？
- 1.2 描述人工智能中的图灵测试，并从现代的角度证明它的有效性。
- 1.3 如何定义人工智能这门科学？人工智能何时诞生？
- 1.4 什么是弱方法？列举 20 世纪 70 年代早期 AI 幻灭的主要原因。
- 1.5 何为专家系统？弱方法和专家系统技术的主要区别是什么？
- 1.6 列举 DENDRAL、MYCIN 和 PROSPECTOR 这些早期专家系统的共同特征。
- 1.7 专家系统的局限性有哪些？
- 1.8 专家系统和人工神经网络之间的区别是什么？
- 1.9 人工神经网络在 20 世纪 80 年代复活的原因是什么？
- 1.10 模糊逻辑的建立前提是什么？模糊集理论是何时出现的？
- 1.11 在基于知识的系统中应用模糊逻辑的主要优势是什么？
- 1.12 合并专家系统、模糊逻辑和神经计算的系统会有哪些优势？

参考文献

- Barto, A.G., Sutton, R.S. and Anderson C.W. (1983). Neurolike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, 834–846.
- Boden, M.A. (1977). *Artificial Intelligence and Natural Man*. Basic Books, New York.
- Broomhead, D.S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks, *Complex Systems*, 2, 321–355.
- Bryson, A.E. and Ho, Y.-C. (1969). *Applied Optimal Control*. Blaisdell, New York.
- Buchanan, B.G., Sutherland, G.L. and Feigenbaum, E.A. (1969). Heuristic DENDRAL: a program for generating explanatory hypotheses in organic chemistry, *Machine Intelligence 4*, B. Meltzer, D. Michie and M. Swann, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 209–254.
- Cook, S.A. (1971). The complexity of theorem proving procedures, *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, New York, pp. 151–158.
- Cowan, J.D. (1990). Neural networks: the early days, *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, ed., Morgan Kaufman, San Mateo, CA, pp. 828–842.
- Cox, E. (1999). *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*, 2nd edn. Academic Press, San Diego, CA.
- Duda, R., Gaschnig, J. and Hart, P. (1979). Model design in the PROSPECTOR

- consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.
- Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.
- Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program, *Machine Intelligence 6*, B. Meltzer and D. Michie, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 165–190.
- Fogel, D.B. (1995). *Evolutionary Computation – Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA.
- Greenblatt, R.D., Eastlake, D.E. and Crocker, S.D. (1967). The Greenblatt Chess Program, *Proceedings of the Fall Joint Computer Conference*, pp. 801–810.
- Grossberg, S. (1980). How does a brain build a cognitive code? *Psychological Review*, 87, 1–51.
- Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.
- Holland, J.H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Perseus Books, New York.
- Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the USA*, 79, 2554–2558.
- Karp, R.M. (1972). Reducibility among combinatorial problems, *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher, eds, Plenum, New York, pp. 85–103.
- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 59–69.
- Kosko, B. (1993). *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion, New York.
- Kosko, B. (1997). *Fuzzy Engineering*. Prentice Hall, Upper Saddle River, NJ.
- Koza, J.R. (1992). *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.
- Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA.
- LeCun, Y. (1988). A theoretical framework for back-propagation, *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hilton and T. Sejnowski, eds, Morgan Kaufmann, San Mateo, CA, pp. 21–28.
- Lighthill, J. (1973). Artificial intelligence: a general survey, *Artificial Intelligence: A Paper Symposium*, J. Lighthill, N.S. Sutherland, R.M. Needham, H.C. Longuest-Higgins and D. Michie, eds, Science Research Council of Great Britain, London.
- McCarthy, J. (1958). Programs with common sense, *Proceedings of the Symposium on Mechanisation of Thought Processes*, vol. 1, London, pp. 77–84.
- McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5, 115–137.
- Medsker, L. and Leibowitz, J. (1994). *Design and Development of Expert Systems and Neural Computing*. Macmillan, New York.
- Minsky, M.L. (1975). A framework for representing knowledge, *The Psychology of Computer Vision*, P. Winston, ed., McGraw-Hill, New York, pp. 211–277.
- Minsky, M.L. and Papert, S.A. (1969). *Perceptrons*. MIT Press, Cambridge, MA.
- Negoita, C.V. (1985). *Expert Systems and Fuzzy Systems*. Benjamin/Cummings, Menlo Park, CA.
- Newell, A. and Simon, H.A. (1961). GPS, a program that simulates human thought, *Lernende Automaten*, H. Billing, ed., R. Oldenbourg, Munich, pp. 109–124.
- Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ.
- Omlin, C.W. and Giles, C.L. (1996). Rule revision with recurrent neural networks, *IEEE Transactions on Knowledge and Data Engineering*, 8(1), 183–188.
- Parker, D.B. (1987). Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order Hebbian learning, *Proceedings of the IEEE 1st International Conference on Neural Networks*, San Diego,

- CA, vol. 2, pp. 593–600.
- Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley, Hoboken, NJ.
- Rechenberg, I. (1965). *Cybernetic Solution Path of an Experimental Problem*. Ministry of Aviation, Royal Aircraft Establishment, Library Translation No. 1122, August.
- Rechenberg, I. (1973). *Evolutionsstrategien – Optimierung Technischer Systeme Nach Prinzipien der Biologischen Information*. Friedrich Frommann Verlag (Günther Holzboog K.G.), Stuttgart–Bad Cannstatt.
- Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, Chicago.
- Rumelhart, D.E. and McClelland, J.L., eds (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, 2 vols. MIT Press, Cambridge, MA.
- Samuel, A.L. (1959). Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development*, 3(3), 210–229.
- Samuel, A.L. (1967). Some studies in machine learning using the game of checkers II – recent progress, *IBM Journal of Research and Development*, 11(6), 601–617.
- Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. John Wiley, New York.
- Shannon, C.E. (1950). Programming a computer for playing chess, *Philosophical Magazine*, 41(4), 256–275.
- Shortliffe, E.H. (1976). *MYCIN: Computer-Based Medical Consultations*. Elsevier Press, New York.
- Turban, E., Sharda, R. and Delen, D. (2010). *Decision Support and Business Intelligent Systems*, 9th edn. Prentice Hall, Englewood Cliffs, NJ.
- Turing, A.M. (1950). Computing machinery and intelligence, *Mind*, 59, 433–460.
- van Melle, W. (1979). A domain independent production-rule system for consultation programs, *Proceedings of the IJCAI 6*, pp. 923–925.
- van Melle, W., Shortliffe, E.H. and Buchanan B.G. (1981). EMYCIN: a domain-independent system that aids in constructing knowledge-based consultation programs, *Machine Intelligence, Infotech State of the Art Report 9*, no. 3.
- Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading, MA.
- Yager, R.R. and Zadeh, L.A., eds (1994). *Fuzzy Sets, Neural Networks and Soft Computing*. Van Nostrand Reinhold, New York.
- Zadeh, L. (1965). Fuzzy sets, *Information and Control*, 8(3), 338–353.
- Zahedi, F. (1993). *Intelligent Systems for Business: Expert Systems with Neural Networks*. Wadsworth, Belmont, CA.

基于规则的专家系统

本章介绍构建知识系统的常用选择：基于规则的专家系统。

2.1 知识概述

在 20 世纪 70 年代，人们最终意识到，要用机器解决智能问题，就必须先知道方案。也就是说，人必须具备特定领域的知识，知道如何做。

知识是什么

知识是对于一个主题或一个领域在理论或实践上的理解，也是所有已知的总和，显然，知识就是力量。那些拥有知识的人称为专家，他们是所在组织中重要的人物。任何一个成功运营的公司都有若干一流的专家，没有他们公司就无法运营下去。

谁可称为专家

任何一个人如果在特定领域有深厚的知识（表现在事实和规则两方面）和强大的实践经验，都被认为是领域专家。领域的范围有所限制。例如，发电机专家也许对变压器仅有泛泛的了解，人寿保险专家或许对不动产保险政策的了解也是有限的。总之，专家能熟练地做别人做不了的事情。

专家如何思考

人类的思维活动是内在的、复杂的，用算法表达不了。好在多数专家能以规则的形式表达知识，从而解决问题。举一个简单的例子，假设你遇到一个外星人，他想过马路。你是否帮得了他？因为有多年过马路的经历，这一方面你是专家。所以你有能力就此教外星人。你会怎么做呢？

你可以告诉外星人，当交通灯变绿时，可以安全地过马路，当交通灯变红时，就必须停止。这是基本规则。可以把你的知识简单地表达为如下句子：

```
IF      the 'traffic light' is green
THEN   the action is go

IF      the 'traffic light' is red
THEN   the action is stop
```

这些用 IF-THEN 表达的句子称为产生式规则或规则。在人工智能学科中，规则是最常用的知识表达的方式，可以定义为 IF-THEN 语句，IF 部分是给定的信息或事实，THEN 部分则是相应的行为。规则用于描述如何解决问题，创建简单，也易于理解。

2.2 知识表达技术——规则

任何规则都包括两部分：IF 部分和 THEN 部分。IF 部分是前项（前提或条件），THEN 部分是后项（结论或行为）。

规则的基本语法是：

```
IF      <前项>
THEN   <后项>
```

一般而言，规则可以有多个前项，这些前项用关键词 AND（合取）、OR（析取）或者 AND、OR 混合使用。不过，最好在同一个规则中避免混合使用合取和析取。

```
IF    <前项 1>
AND   <前项 2>
      ⋮
AND   <前项 n>
THEN  <结论>

IF    <前项 1>
OR     <前项 2>
      ⋮
OR     <前项 n>
THEN  <结论>
```

规则的结论部分也可以由多个从句组成：

```
IF    <前项>
THEN  <结论 1>
      <结论 2>
      ⋮
      <结论 m>
```

规则的前项部分包括两部分：对象（语言对象）和值。以过马路为例，语言对象是“交通灯”，可以取值“绿”或“红”。对象和值用操作符连接。操作符识别对象并进行赋值。像 is、are、is not、are not 等操作符为语言对象赋予符号值。在专家系统中，也可以使用数学操作符定义数值型对象，并赋予数值。例如：

```
IF    'age of the customer' < 18
AND   'cash withdrawal' > 1000
THEN  'signature of the parent' is required
```

规则的后项部分与前项部分类似，也通过操作符连接对象和值。操作符为语言对象赋值。以过马路为例，如果“交通灯”的值是“绿”，第一条规则会将语言对象的行为赋值为“前进”。后项中也可以用数值型对象和简单的算术表达式：

```
IF    'taxable income' > 16283
THEN  'Medicare levy' = 'taxable income' * 1.5 / 100
```

关系、建议、指示、策略和启发式方法也可以用规则表达：

关系

```
IF    the 'fuel tank' is empty
THEN  the car is dead
```

建议

```
IF    the season is autumn
AND   the sky is cloudy
AND   the forecast is drizzle
THEN  the advice is 'take an umbrella'
```

指示

```
IF    the car is dead
AND   the 'fuel tank' is empty
THEN  the action is 'refuel the car'
```

策略

```
IF    the car is dead
THEN  the action is 'check the fuel tank';
      step1 is complete

IF    step1 is complete
AND   the 'fuel tank' is full
THEN  the action is 'check the battery';
      step2 is complete
```

启发式方法

```
IF      the spill is liquid
AND    the 'spill pH' < 6
AND    the 'spill smell' is vinegar
THEN   the 'spill material' is 'acetic acid'
```

2.3 专家系统研发团队的主要参与者

当人类专家提供了知识后，我们就能够把知识输入计算机，期待着在特定领域的计算机能够做一个智能的助手，或者像专家那样解决问题，并希望计算机能够整合新知识，并以易读、易懂的方式表达知识，或用自然语言而非人工编程语言来处理简单的句子。最后，我们还希望计算机解释得出一个结论的原因。换言之，我们需要构建专家系统——能在狭窄问题领域具有专家水平的计算机程序。

基于规则的系统是最常见的专家系统。已有大量的系统被成功应用于商业、工程、医药、地理、电力系统以及采矿，也有很多公司生产和销售用于开发基于规则的专家系统的软件，即个人电脑上的专家系统框架。

为开发基于规则的系统的专家系统框架越来越普遍，其主要优势在于系统开发员可以把注意力放在知识本身，而不是学习一门编程语言。

什么是专家系统框架

专家系统框架（expert system shell）是不含知识的专家系统。用户的全部任务就是以规则形式加入知识，并提供解决问题的相关数据。

现在看一下构建专家系统需要哪些人员，这些人员需要具备什么能力？

一般地，专家系统开发团队需要 5 个成员：领域专家、知识工程师、程序员、项目经理和终端用户。专家系统能否开发成功完全取决于成员间能否很好地合作。图 2.1 总结了开发团队中成员的基本关系。

领域专家是在特定区域或领域具有渊博知识和丰富经验的人。他们在特定领域具有最好的经验，这些经验会被输入到专家系统中。所以，专家必须能够传达知识，乐意加入开发团队，并在项目上有足够的时间。在专家系统开发团队中，领域专家是最重要的成员。

知识工程师是善于设计、构建、测试专家系统的人，负责为系统选择合适的任务，并就如何解决特定问题和领域专家进行互动。通过互动，知识工程师会明确专家如何解决事实和规则，以及怎样在专家系统中表达。继而选择特定的开发软件或专家系统框架，或选择编程语言以将知识编码（有时知识工程师亲自编码知识）。之后，知识工程师负责测试、修改专家系统，并将系统集成到工作平台。可以看出，知识工程师要负责从系统的最初设计到最终运行的整个流程。项目完成后，也许还要参与系统的维护。

程序员负责编程，用计算机能接受的术语描述领域知识。他不仅需要精通人工智能语言的符号编程，像 LISP、PROLOG 和 OPS5 等，也要具备不同类型专家系统框架的应用经验。此外，程序员还要会常用编程语言，如 C、Pascal、FORTRAN 和 Basic。如果使用专家系统框架进行开发，知识工程师就能够将知识编码到系统中，就不需要程序员了。当不能使用专家系统框架时，

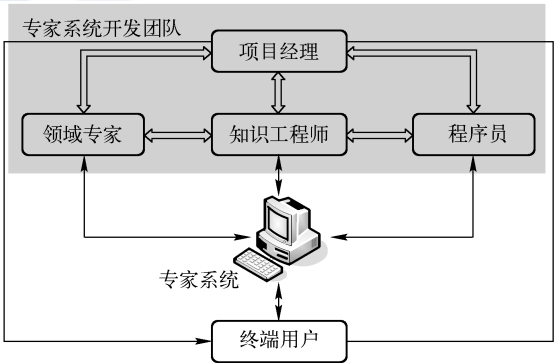


图 2.1 专家系统开发团队的主要成员

就需要程序员设计知识和数据表达结构（知识库和数据库）、控制结构（推理引擎），以及对话结构（用户界面）。程序员可能也要参与测试专家系统。

项目经理是开发团队的领导，负责项目的进程，确保按计划进行，并与专家、知识工程师、程序员、终端用户进行互动。

终端用户，也称为用户，是使用专家系统的人，可能是一个判定火星表面土壤的化学结构的分析化学家（Feigenbaum et al.，1971），或诊断传染性血液病的初级医生（Shortliffe，1976），或勘探新矿床的勘探地质学家（Duda et al.，1979），或紧急情况下需要建议的电力系统操作员（Negnevitsky，2008）。这些终端用户各有所需，所以系统最终能否被认可依赖于用户的满意度。用户的满意度不仅包括系统功能，还包括操作是否舒适。所以系统中用户界面的设计至关重要。因此，终端用户的建议至关重要。

当聚齐 5 个成员时，就可以开发系统了。不过，现在的许多专家系统是在个人电脑上使用专家系统框架开发的，不需要程序员，或许也不需要知识工程师。对于微型专家系统，项目经理、知识工程师、程序员甚至专家可能是同一个人。在开发大型专家系统时，就需要上述所有的成员了。

2.4 基于规则的专家系统的结构

在 20 世纪 70 年代早期，来自卡内基 - 梅隆大学的 Newell 和 Simon 提出了产生式系统模型，为现代的基于规则的专家系统奠定了基础（Newell and Simon，1972）。产生式模型的思路是，对于一个用相关信息表达的指定问题，人类运用知识（用产生式规则表达）能够解决。产生式规则存储在长期存储器中，问题相关的信息或事实存储在短期存储器中。图 2.2 是产生式系统模型和基于规则的专家系统的基本结构。

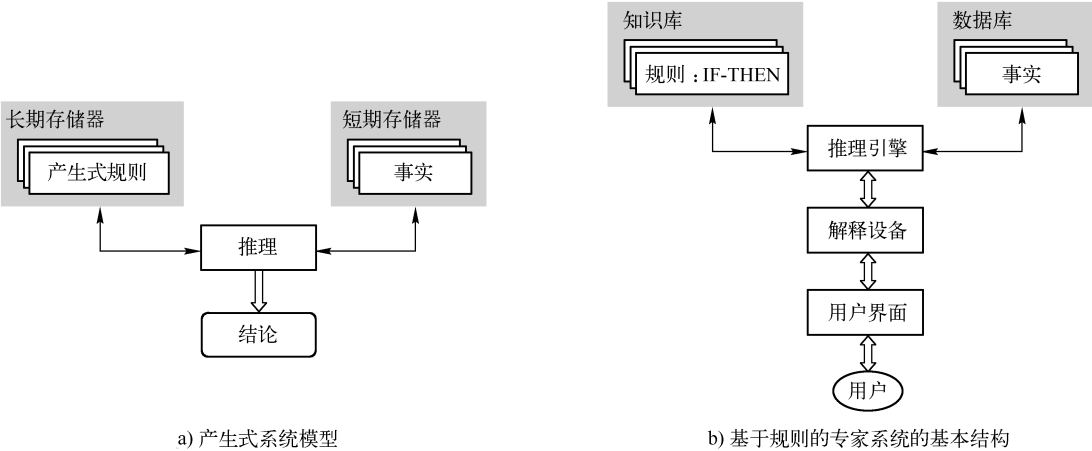


图 2.2 产生式系统模型和基于规则的专家系统的基本结构

基于规则的专家系统由 5 个部分组成：知识库、数据库、推理引擎、解释设备和用户界面。

知识库包含解决问题相关的领域知识。在基于规则的专家系统中，知识用一组规则来表达。每一条规则表达一个关系、建议、指示、策略或启发式方法，具有 IF（条件） THEN（行为）结构。当规则的条件被满足时，触发规则，继而执行行为。

数据库包含一组事实，用于匹配存储在知识库中的 IF（条件）部分。

推理引擎执行推理，专家系统由此找到解决方案。推理引擎链接知识库中的规则和数据库中的事实。

用户使用解释设备查看专家系统怎样得出解决方案的过程，以及为什么需要特定事实。专家系统必须能够解释推理并证明所给的建议、分析或结论。

用户界面是实现用户（查询问题解决方案）和专家系统之间交流的途径，这一途径必须有实际意义并尽可能地友好。

这5个部分对于任何基于规则的专家系统来说都是不可或缺的，它们共同构成了专家系统的核心，除此之外也会有其他附加组件。

外部接口允许专家系统融合外部数据文件和以常用语言编写的程序，如 C、Pascal、FORTRAN 和 Basic。图 2.3 是基于规则的专家系统的完整结构。

开发者接口一般包括知识库编辑器、调试工具，以及输入/输出设备。

任何专家系统框架都会提供简单的文本编辑器，用于输入和调整规则、检查格式和拼写。许多专家系统也包括记录设备，用于监测知识工程师或专家所做的修改。如果规则被改动，编辑器将自动存储改动日期和做改动的人，以便后面参考。当有多个知识工程师和专家都有权进入知识库并做修改时，记录设备尤其重要。

调试工具一般包含跟踪设备和断点包。跟踪设备提供程序执行过程中被触发的规则列表。使用断点包则能提前告诉系统哪里该中断，以备知识工程师或专家即时查看数据库中的当前值。

多数专家系统还提供输入/输出设备，例如运行时知识获取器，以便运行中的专家系统获取数据库之外的必需信息。当知识工程师或专家输入所需信息后，系统接着往下运行。

总之，开发者接口、知识获取设备使得领域专家能够直接将知识输入专家系统，以减少打扰知识工程师的次数。

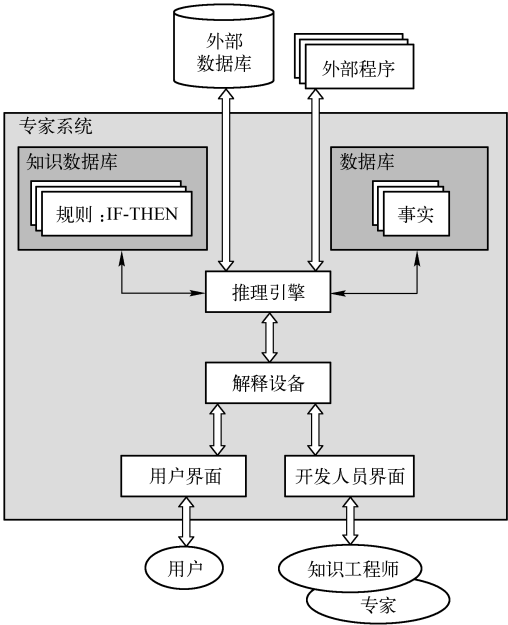


图 2.3 基于规则的专家系统的完整结构

2.5 专家系统的基本特征

专家系统用于在狭窄的特定领域以专家级的水平解决问题。因此，专家系统的最主要特征就在于高质量的性能。无论系统能多快地解决问题，如果结果错误，用户是不会满意的。另一方面，解决速度也很重要。最精确的决策或诊断如果来的太慢，也许也会无用，例如在病人死亡或核反应堆爆炸等紧急情况时。专家使用经验法则或启发式方法，通过自己的经验和对问题的理解来寻找问题的解决捷径。专家系统也应该像人类专家那样，运用启发式方法来引导推理并降低解决方案的搜索空间。

拥有解释功能是专家系统独具的特征，专家系统通过这一功能回顾推理过程，并对其决策进行解释。所谓解释是跟踪整个解决过程中被触发的规则。这当然是一个简单的说法。事实上，由于系统不具备对该领域的最起码的理解，真正的、像人那样的解释还不现实。尽管一系列被触发的规则并不能证明一个结论，我们仍然可以为每一条规则，至少为知识库中的高级规则，关联上用文本表达的恰当的基本规律。这也许是目前解释功能所能达到的程度。不过，解释功能对有些专家系统来说不是必需的。例如，为专家设计的科学系统不需要提供大量解释，因为结论对

专家而言是自解释的，有简单的规则 – 跟踪机制就足够了。另一方面，用于决策支持的专家系统则需要完整的、有意义的解释，毕竟错误结论的代价很高。

在解决问题时，专家系统使用符号推理。符号被用来表示不同类型的知识，如事实、概念和规则。构建专家系统的目的是处理知识，因此能够轻易地处理定性数据，这点不同于处理数值数据的传统程序。

传统程序通过算法处理数据，换言之，它是定义好的逐步的操作序列。一个算法总按同一顺序执行同样的操作，每次都得到精确解。传统程序不会出错，只是程序员有时出错。专家系统则与传统程序不同，因为它并不按照特定的步骤序列执行，允许不精确的推理，也能够处理不完整、不确定的模糊数据。

专家系统会出错吗

再聪明的专家也是人，也会出错。对于要达到人类专家水平的专家系统，也应该容许出错。尽管我们知道专家的判断有时错误，但是我们仍然信赖专家。同样，尽管在多数情况下专家系统所得的结论都是可信赖的，有时依然会出错，我们要认识到这一点。

这意味着传统程序优于专家系统吗

理论上，传统程序提供的答案总是正确的。但要记住传统程序仅能处理数据完整、准确的问题。当数据不完整或包含错误时，传统程序要么无法应对，要么给出错误的答案。相比之下，专家系统能够识别出不完整或者模糊的信息，在这种情况下仍能给出合理的结论。

专家系统有别于传统程序的另一个特征是，知识与处理过程相分离（知识库和推理引擎是两部分）。而传统程序中知识和处理知识的控制结构是混合在一起的，这使得我们难以理解和修改程序代码，毕竟对代码的修改既影响知识，又影响知识的处理。在专家系统中，知识与处理机制被明确地划分开，在构建和维护专家系统时就容易得多。如果使用专家系统框架，知识工程师或者专家只需往知识库中添加知识即可。每一条新规则会带来新知识，专家系统也随之更聪明。通过改变、删减规则，能轻易地修改系统。

专家系统的上述特征使专家系统有别于传统程序和人类专家。表 2.1 是三者之间的对比。

表 2.1 专家系统、传统程序及人类专家的对比

人 类 专 家	专 家 系 统	传 统 程 序
在狭窄领域，以经验法则或启发式方法的知识形式来解决问题	在狭窄领域，以规则形式表达知识，使用符号推理解决问题	处理数据，使用由一系列操作组成的算法，来解决一般的数值问题
在人脑中，知识以可编译的形式存在	提供知识与处理过程明确分离的机制	知识与处理知识的控制结构没有分离
能够解释推理过程并提供细节	在解决问题过程中跟踪被触发的规则，能够解释怎样得出的结论，以及为什么需要特定数据的原因	对怎样得出结论和需要数据的原因都不做解释
使用不精确的推理，能处理不完整、不确定和模糊的信息	允许不精确的推理，能处理不完整、不确定和模糊的数据	仅能在数据完整、确定的情况下处理问题
当信息不完整或模糊时，可能会出错	当数据不完整或模糊时，可能会出错	当数据不完整或模糊时，要么无能为力，要么出错
通过多年的学习和实践训练，可以提高解决问题的能力。不过这一过程缓慢、低效、代价高	通过向知识库加入新规则或调整老规则，可以提高解决问题的能力。当需要新知识时，很容易实现调整	通过改变程序代码可以提高解决问题的能力。由于对代码的改变将同时影响知识和处理过程，改变起来很困难

2.6 前向链接和后向链接推理技术

在基于规则的专家系统中,领域知识用一组IF-THEN产生式规则来表示,数据用当时情境

下的一组事实来表示。推理引擎将存储在知识库中的规则与数据库中的事实相匹配，如果一条规则的 IF（条件）部分与事实匹配，即触发该规则，执行 THEN（行为）部分。被触发的规则可能由于添加新事实而改动事实集合，如图 2.4 所示。数据库和知识库中的字母表示状态或概念。

通过匹配规则的 IF 部分和事实，形成推理链。推理链表明了专家系统如何运用规则得出结论。下面通过一个简单的例子来解释链式推理技术。

假设数据库最初包含 5 个事实：A、B、C、D 和 E，知识库最初包含 3 条规则：

- Rule 1: IF Y is true
AND D is true
THEN Z is true
- Rule 2: IF X is true
AND B is true
AND E is true
THEN Y is true
- Rule 3: IF A is true
THEN X is true

图 2.5 中的推理链表明了专家系统运用规则最终推出事实 Z 的过程。规则 3 最早被激活，由事实 A 推出了新事实 X。之后规则 2 被激活，由初始事实 B 和 E，加上事实 X，共同推出事实 Y。最后，规则 1 被激活，由初始事实 D 和刚确定的事实 Y 推出事实 Z。

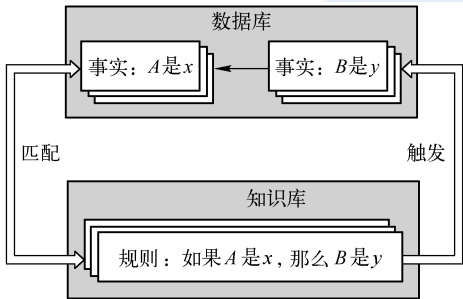


图 2.4 推理引擎中的匹配 - 触发循环过程

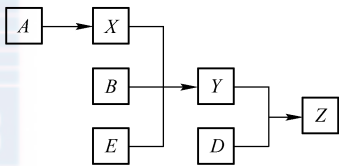


图 2.5 推理链的例子

专家系统能够用推理链来解释怎样得出的结论，这是解释设备的必要部分。推理引擎要决定何时激活哪条规则。选择规则时，有两个主要方法：前向链接和后向链接（Waterman and Hayes-Roth，1978）。

2.6.1 前向链接

上述例子正是运用了前向链接。现在来具体地分析这一技术。我们先将规则重写为如下形式：

- Rule 1: $Y \& D \rightarrow Z$
- Rule 2: $X \& B \& E \rightarrow Y$
- Rule 3: $A \rightarrow X$
- 箭头表明规则的 IF 部分和 THEN 部分。另外，再添加两条规则：
- Rule 4: $C \rightarrow L$
- Rule 5: $L \& M \rightarrow N$

图 2.6 显示了这一组规则按前向链接的运作流程。

前项链接是数据驱动（data-driven）的推理技术。从已知数据开始展开推理。每一次只执行顶端的一条规则。当有规则被触发时，就有新事实加入数据库。任何规则只能被执行一次。当没

有规则可触发时，匹配 - 触发循环终止。

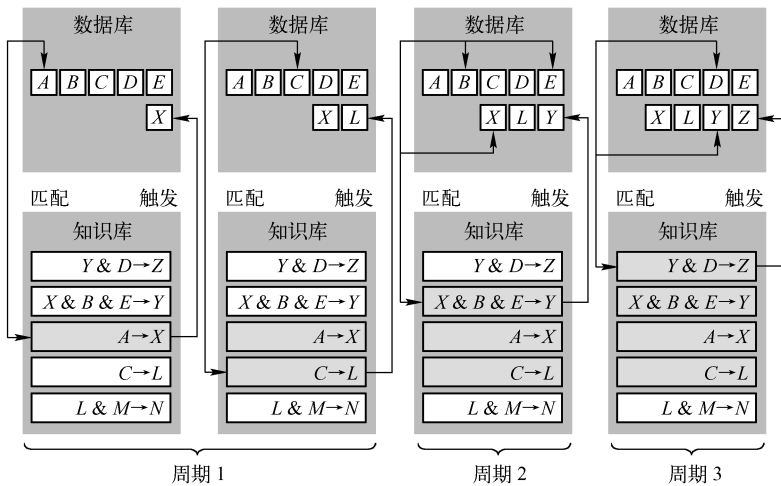


图 2.6 前向链接

第一轮中，只有两个规则，规则 3： $A \rightarrow X$ 和规则 4： $C \rightarrow L$ ，与数据库中的事实匹配。规则 3： $A \rightarrow X$ 处于顶端，就先被触发。规则 3 的 IF 部分与数据库中的事实 A 相匹配，执行 THEN 部分后，将新事实 X 加入数据库。之后，规则 4： $C \rightarrow L$ 被触发，新事实 L 加入数据库。

第二轮中，由于数据库中已有事实 B、E 和 X。规则 2： $X \& B \& E \rightarrow Y$ 最先被触发，产生事实 Y，并被放入数据库。这导致第三轮中规则 1： $Y \& D \rightarrow Z$ 被执行，生成新事实 Z。至此，匹配 - 激活循环终止。由于规则 5： $L \& M \rightarrow N$ 不匹配所有数据库中的事实，规则 5 无法被触发。

前向链接是搜集信息并推出信息的技术。不过，在前向链接中，许多被触发的规则也许与问题目标无关。在上面的例子中，目标是推出事实 Z。我们仅有 5 条规则，其中 4 条规则被触发。规则 4： $C \rightarrow L$ 虽与事实 Z 无关，也同样被触发了。一个真正的基于规则的专家系统可能有成百上千条规则，许多规则被触发后虽然能推出有效的新事实，但可能与目标无关。所以，如果只需推出一个特定的事实，前向链接推导技术或许效率很低。

这种情况下，后向链接技术就比较合适。

2.6.2 后向链接

后向链接是目标驱动的推理技术。在后向链接中，专家系统有目标（一个假设的答案），推理引擎的任务是找出证明目标的论据。首先，在知识库中搜寻含有目标的规则，即 THEN 部分包含的目标规则。如果找到这种规则，在数据库中也有匹配的事实，就触发规则并证明目标。不过这种情况很少见。所以，推理引擎就暂不考虑这类规则（将规则压栈），要建立新目标，即子目标，以证明压栈规则的 IF 部分。接下来，再次查找知识库中能证明子目标的规则。推理引擎不断将规则压栈，直到知识库中的所有规则都不能证明子目标。

图 2.7 以上述例子为例，显示了后向链接的工作原理。

在第 1 个周期，推理引擎尝试推出事实 Z。通过查找知识库来寻找 THEN 部分包含事实 Z 的规则。推理引擎找到了规则 1： $Y \& D \rightarrow Z$ ，将它压栈。规则 1 的 IF 部分包含事实 Y 和 D，所以下面是确立事实 Y 和 D。

在第 2 个周期，推理引擎安装子目标：事实 Y，并试图确立它。通过检查知识库，发现其中

没有事实 Y 。因此需要寻找 THEN 部分包含事实 Y 的规则。推理引擎发现规则 2: $X \& B \& E \rightarrow Y$ 满足条件, 将它压栈。规则 2 的 IF 部分包括事实 X 、 B 和 E , 接下来同样需要建立这些事实。

在第 3 个周期, 推理引擎安装新的子目标: 证实事实 X 。推理引擎通过查找数据库, 发现不包含 X , 就去查找能推出 X 的规则。找到了规则 3: $A \rightarrow X$, 将它压栈。接下来需要证实事实 A 。

在第 4 个周期, 推理引擎发现 A 在数据库中。规则 3: $A \rightarrow X$ 被触发, 推出了新事实 X 。

在第 5 个周期, 推理引擎要证实子目标 Y , 再次试图执行规则 2: $X \& B \& E \rightarrow Y$ 。由于事实 X 、 B 和 E 都在数据库中, 规则 2 被触发, 推出了新事实 Y , 将 Y 加入数据库。

在第 6 个周期, 系统转向规则 1: $Y \& D \rightarrow Z$, 尝试证实初始目标 Z 。由于规则 1 的 IF 部分与数据库中的事实相匹配, 规则 1 被执行, 初始目标得以证实。

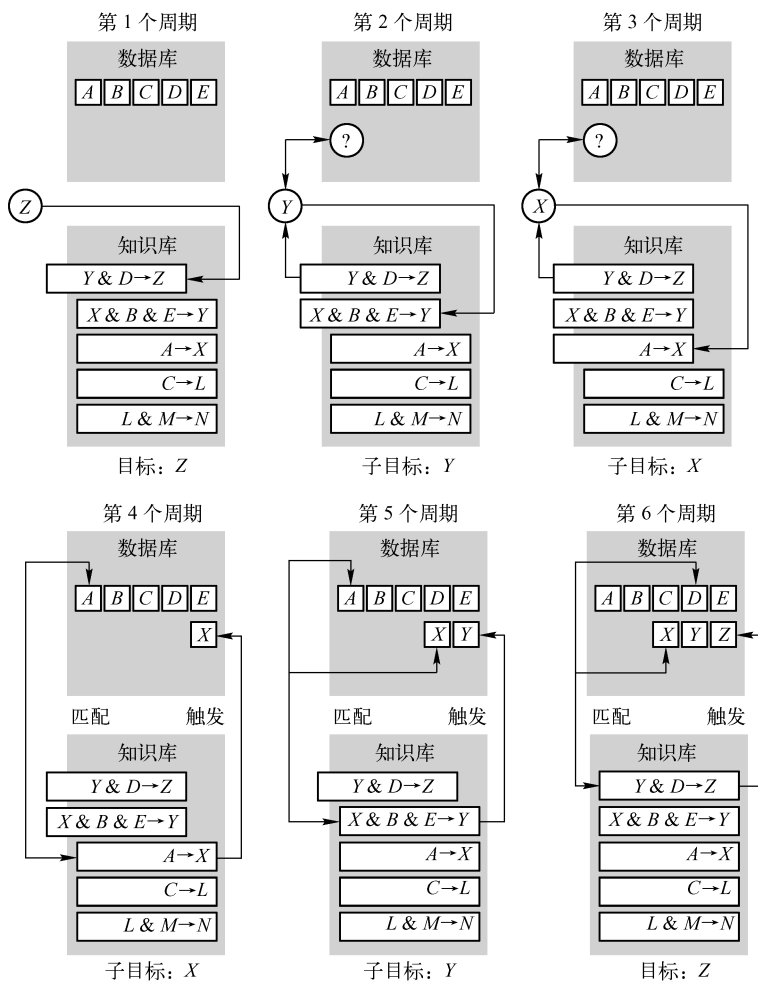


图 2.7 后向链接

现在来对比图 2.6 和图 2.7。可以看出, 前向链接中触发了 4 条规则, 而后向链接中只触发了 3 条规则。这个简单的例子说明了当需要证实一个特定事实时 (例子中是 Z), 后向链接更为有效。前向链接中, 在推理过程开始前就确定了数据, 不需用户额外输入。在后向链接中, 确立目标后, 只需要那些支持推理的数据, 有时也需要用户输入一些数

据库中没有的事实。

如何选择前向链接和后向链接

答案是分析领域专家是如何解决问题的。如果专家事先需要搜集信息，无论结论是什么都试着去推理，这时应选择前向链接推理技术。如果专家从一个假设的结论触发，尝试着找出支持结论的论据，则应选择后向链接推理技术。

前向链接对于设计能够分析和解释的专家系统来说，是理所当然的方法。例如 DEN-DRAL，这款专家系统基于大量质谱数据来确定未知土壤的分子结构（Feigenbaum et al.，1971），就使用了前向链接技术。多数使用后向链接的专家系统多用于诊断性工作。例如，诊断传染性血液病的医用专家系统 MYCIN（Shortliffe，1976），就使用了后向链接技术。

能否将前项、后向链接相结合

许多专家系统框架同时使用了前项、后向链接推理技术，所以知识工程师就不必二选一。不过，基本的推理机制一般采用后向链接。只有当确立新事实时，才使用前向链接以最大程度地利用新数据。

2.7 MEDIA ADVISOR：基于规则的专家系统实例

接下来通过一个简单的基于规则的专家系统进一步说明上述概念。我们选择专家系统框架 Leonardo 来构建一个名为 MEDIA ADVISOR 决策支持系统，用于为实习生推荐平台，该平台根据实习生的工作制订培训计划。例如，如果实习生是一个负责维护水压系统的机械技术员，车间就是合适的平台，在那里可以学习水压组件的工作原理、如何检测水压故障，以及如何对水压系统做简单修复。如果实习生是评估投保单的员工，培训计划中就需包括特定相关任务的讲座，以及实习生评估实际申请的教程。对于复杂任务，实习生可能会出错，培训计划就应包括对实习生绩效的反馈。

知识库

/* MEDIA ADVISOR: a demonstration rule-based expert system

Rule: 1

```
if    the environment is papers
or    the environment is manuals
or    the environment is documents
or    the environment is textbooks
then  the stimulus_situation is verbal
```

Rule: 2

```
if    the environment is pictures
or    the environment is illustrations
or    the environment is photographs
or    the environment is diagrams
then  the stimulus_situation is visual
```

Rule: 3

```
if    the environment is machines
or    the environment is buildings
or    the environment is tools
then  the stimulus_situation is 'physical object'
```

Rule: 4

```
if    the environment is numbers
or    the environment is formulas
or    the environment is 'computer programs'
then  the stimulus_situation is symbolic
```



```
Rule: 5
if    the job is lecturing
or    the job is advising
or    the job is counselling
then  the stimulus_response is oral

Rule: 6
if    the job is building
or    the job is repairing
or    the job is troubleshooting
then  the stimulus_response is 'hands-on'

Rule: 7
if    the job is writing
or    the job is typing
or    the job is drawing
then  the stimulus_response is documented

Rule: 8
if    the job is evaluating
or    the job is reasoning
or    the job is investigating
then  the stimulus_response is analytical

Rule: 9
if    the stimulus_situation is 'physical object'
and   the stimulus_response is 'hands-on'
and   feedback is required
then  medium is workshop

Rule: 10
if    the stimulus_situation is symbolic
and   the stimulus_response is analytical
and   feedback is required
then  medium is 'lecture – tutorial'

Rule: 11
if    the stimulus_situation is visual
and   the stimulus_response is documented
and   feedback is not required
then  medium is videocassette

Rule: 12
if    the stimulus_situation is visual
and   the stimulus_response is oral
and   feedback is required
then  medium is 'lecture – tutorial'

Rule: 13
if    the stimulus_situation is verbal
and   the stimulus_response is analytical
and   feedback is required
then  medium is 'lecture – tutorial'

Rule: 14
if    the stimulus_situation is verbal
and   the stimulus_response is oral
and   feedback is required
then  medium is 'role-play exercises'

/* The SEEK directive sets up the goal of the rule set
seek medium
```

对象

MEDIA ADVISOR 使用 6 个语言对象：*environment*、*stimulus_situation*、*job*、*stimulus_response*、*feedback* 和 *medium*。各个对象都可以在各自值域内取一个值（例如，*environment* 可取的值有

papers、*manuals*、*documents*、*textbooks*、*pictures*、*illustrations*、*photographs*、*diagrams*、*machines*、*buildings*、*tools*、*numbers*、*formulas*、*computer programs*)。对象和它的值构成事实 (例如, 如果 *environment* is *machines* 和那么 *job* is *repairing*)。所有的事实都存在数据库中。

表 2.2 MEDIA ADVISOR 语言对象及其可取值

对 象	可 取 值	对 象	可 取 值
environment	papers manuals documents textbooks pictures illustrations photographs diagrams machines buildings tools numbers formulas computer programs	job	lecturing advising counselling building repairing troubleshooting writing typing drawing evaluating reasoning investigating
		stimulus_response	oral hands-on documented analytical
stimulus_situation	verbal visual physical object symbolic	feedback	required not required

选项

基于规则的专家系统的最终目标是根据数据找出问题的解决方案。在 MEDIA ADVISOR 中, 解决方案是从 4 个选项中选择 一个平台。

medium is workshop
medium is 'lecture – tutorial'
medium is videocassette
medium is 'role-play exercises'

对话

在如下的对话中, 系统要求用户输入所需数据以解决问题 (问题有环境、工作和反馈三方面)。根据用户提供的答案 (如下面箭头所示), 专家系统应用规则来推出 *stimulus_situation* is *physical object*, 以及 *the stimulus_response* is *hands-on* 等结论, 再根据规则 9 选择平台。

What sort of environment is a trainee dealing with on the job?
⇒ **machines**

Rule: 3
if the environment is machines
or the environment is buildings
or the environment is tools
then the stimulus_situation is 'physical object'

In what way is a trainee expected to act or respond on the job?
⇒ **repairing**

Rule: 6
if the job is building
or the job is repairing
or the job is troubleshooting
then the stimulus_response is 'hands-on'

Is feedback on the trainee's progress required during training?
⇒ **required**

Rule: 9
if the stimulus_situation is 'physical object'
and the stimulus_response is 'hands-on'
and feedback is required
then medium is workshop

推理技术

Leonardo 系统中标准的推理技术是后向链接技术，偶尔使用前向链接，它在处理可获取的信息时最有效。Leonardo 用户也能关掉后向链接或前向链接，这样便能分别研究这两种技术。

前向链接是数据驱动的推理技术，需要预先提供数据。假设：

the environment is **machines**
 'environment' instantiated by user input to 'machines'

the job is **repairing**
 'job' instantiated by user input to 'repairing'

feedback is **required**
 'feedback' instantiated by user input to 'required'

将会发生下述过程：

Rule: 3 fires 'stimulus_situation' instantiated by Rule: 3 to 'physical object'
Rule: 6 fires 'stimulus_response' instantiated by Rule: 6 to 'hands-on'
Rule: 9 fires 'medium' instantiated by Rule: 9 to 'workshop'
No rules fire stop

后向链接是目标驱动的推理技术，需要预先假定结果（目标）。假设目标是“‘medium’ is ‘workshop’”，确立以下目标：

Pass 1	
Trying Rule: 9	Need to find object 'stimulus_situation'
Rule: 9 stacked	Object 'stimulus_situation' sought as 'physical object'
Pass 2	
Trying Rule: 3	Need to find object 'environment'
Rule: 3 stacked	Object 'environment' sought as 'machines'
ask environment	
⇒ machines	'environment' instantiated by user input to 'machines'
Trying Rule: 3	'stimulus_situation' instantiated by Rule: 3 to 'physical object'
Pass 3	
Trying Rule: 9	Need to find object 'stimulus_response'
Rule: 9 stacked	Object 'stimulus_response' sought as 'hands-on'
Pass 4	
Trying Rule: 6	Need to find object 'job'
Rule: 6 stacked	Object 'job' sought as 'building'
ask job	
⇒ repairing	'job' instantiated by user input to 'repairing'
Trying Rule: 6	'stimulus_response' instantiated by Rule: 6 to 'hands-on'
Pass 5	
Trying Rule: 9	Need to find object 'feedback'
Rule: 9 stacked	Object 'feedback' sought as 'required'
ask feedback	
⇒ required	'feedback' instantiated by user input to 'required'

Trying Rule: 9 'medium' instantiated by Rule: 9 to 'workshop'

medium is workshop

用树形图来映射与专家系统的咨询会话很直观。图 2.8 是 MEDIA ADVISOR 的树形图。目标是根结点，当系统启动时，推理引擎搜寻目标的值。

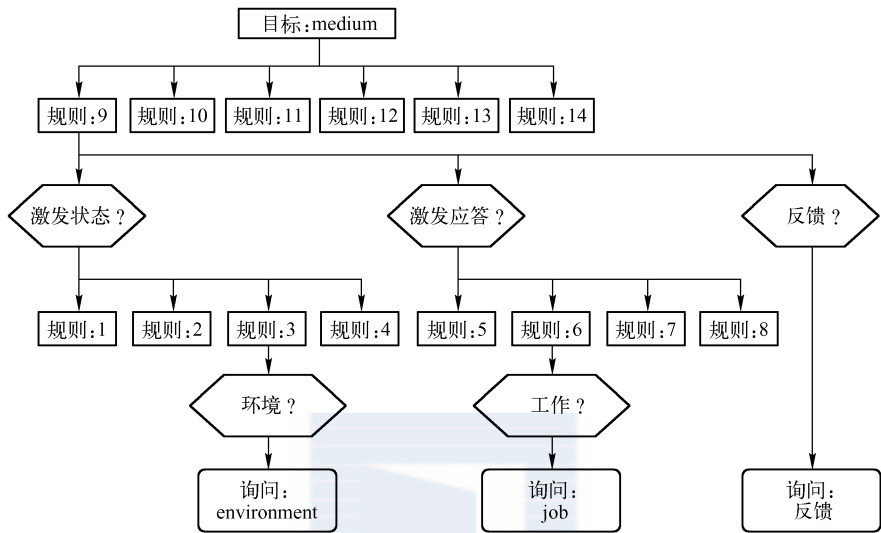


图 2.8 MEDIA ADVISOR 的树形图

MEDIA ADVISOR 能处理任何情况吗

当我们使用专家系统一段时间后，会发现可选项不能涵盖所有情况。例如，可能出现下面的对话。

What sort of environment is a trainee dealing with on the job?
⇒ **illustrations**

In what way is a trainee expected to act or respond on the job?
⇒ **drawing**

Is feedback on the trainee's progress required during training?
⇒ **required**

I am unable to draw any conclusions on the basis of the data.

所以，MEDIA ADVISOR 在现阶段不能处理上面这些情况。幸运的是，通过添加更多规则能轻易地扩展专家系统，直到最终符合用户的需求。

2.8 冲突消解

在本章的开始，我们举过过马路的例子，那是两条规则，现在我们增加第三条规则，便得到了下面的规则集：

- Rule 1:
IF the 'traffic light' is green
THEN the action is go
- Rule 2:
IF the 'traffic light' is red
THEN the action is stop
- Rule 3:
IF the 'traffic light' is red
THEN the action is go

将会发生什么

推理引擎将规则的 IF（条件）部分与数据库中的数据相比较，当条件满足时触发规则。由于一条规则被触发可能影响到其他规则，推理引擎每次只允许触发一条规则。过马路的例子有两条规则，规则 2 和规则 3，它们的 IF 部分相同。当它们的条件被满足时，两条规则都符合触发的条件，从而组成冲突集。当多个规则在一个周期内都满足触发条件时，需要从中选择一条规则，这个方法称为冲突消解（conflict resolution）。

当交通灯变红时，该执行哪条规则

如果用前向链接，那么两条规则都会被触发。规则 2 由于顺序上的优先，先被触发，执行 THEN 部分，语言对象“action”的值是“stop”。“交通灯变红”这一事实仍在数据库中，规则 3 由于满足条件也被触发，相应的对象“action”的新值是“go”。通过这个简单的例子，可以看出当使用前向链接时，规则的顺序很关键。

怎样消解冲突

消解冲突的显而易见的策略是确立目标，当目标实现后不再触发规则。在过马路的例子中，目标是为语言对象“action”确立值。当专家系统为“action”赋值后，目标实现，系统终止。因此，当交通灯变红时，触发规则 2，“action”的值是“stop”，至此系统终止。在这个例子中，专家系统的结论是对的。但如果调换两个规则，结论就错了。这说明知识库中的规则顺序仍然至关重要。

存在其他消解办法吗

还有其他办法（Shirai and Tsuji, 1985；Brachman and Levesque, 2004；Giarratano and Riley, 2004）。

- 触发优先权最高的规则。在简单的应用场景中，合理地排列知识库中的规则就确定了优先权。这种策略一般适于规则数在 100 条左右的系统。但在某些应用中，必须按照重要程度处理数据。例如，在一个医疗咨询中（Durkin, 1994），按照如下方式设置优先权：

```
Goal 1. Prescription is? Prescription

RULE 1 Meningitis Prescription1
(Priority 100)
IF      Infection is Meningitis
AND     The Patient is a Child
THEN    Prescription is Number_1
AND     Drug Recommendation is Ampicillin
AND     Drug Recommendation is Gentamicin
AND     Display Meningitis Prescription1

RULE 2 Meningitis Prescription2
(Priority 90)
IF      Infection is Meningitis
AND     The Patient is an Adult
THEN    Prescription is Number_2
AND     Drug Recommendation is Penicillin
AND     Display Meningitis Prescription2
```

- 激发最具体的规则。这个方法也称为最长匹配策略，其依据的假设是具体规则比一般规则处理更多的信息。例如：

```
Rule 1:
IF      the season is autumn
AND     the sky is cloudy
AND     the forecast is rain
THEN    the advice is 'stay home'

Rule 2:
IF      the season is autumn
THEN    the advice is 'take an umbrella'
```


如果“季节是秋季、多云、预报有雨”这些条件满足，由于规则1的前项比规则2更具体，规则1被触发。如果仅满足“季节是秋季”，规则2被触发。

- 触发那些在数据库中最近加入的数据的规则。这个方法依于数据库中每个事实附带的时间标签。专家系统优先触发冲突集中最新加入数据的规则。例如：

```
Rule 1:
IF    the forecast is rain  [08:16 PM 11/25/96]
THEN  the advice is 'take an umbrella'

Rule 2:
IF    the weather is wet   [10:18 AM 11/26/96]
THEN  the advice is 'stay home'
```

假定这两条规则的 IF 部分都与数据库中的事实匹配。由于“weather is wet”这一事实进入数据库的时间晚于“forecast is rain”，规则2被触发。这个方法尤为适于数据库不断更新的实时应用环境。

以上的冲突消解方法比较简单，也易于实施。多数情况下，这些方法足够用了。但当程序越来越庞大、复杂时，知识工程师管理和检查知识库中的规则的任务就会逐渐困难。必须由专家系统分担任务，并理解系统本身的行为。

为了提高系统性能，我们需要为系统提供关于其所拥有的知识的信息，也就是元知识（metaknowledge）。

元知识可以简单地定义为关于知识的信息，是在专家系统中使用和控制领域知识的信息（Waterman, 1986）。在基于规则的专家系统中，用元规则（metarule）来表示元知识。元规则决定专家系统中具体任务的规则使用策略。

元知识的起源是什么

知识工程师将领域专家的知识传达到专家系统，学习如何使用问题相关的规则，逐渐在头脑里形成新知识体，即关于专家系统行为的信息。这个新知识，即元知识，很大程度上独立于领域。例如：

```
Metarule 1:
Rules supplied by experts have higher priorities than rules supplied by
novices.

Metarule 2:
Rules governing the rescue of human lives have higher priorities than rules
concerned with clearing overloads on power system equipment.
```

专家系统会理解和使用元规则吗

一些专家系统为元规则提供了单独的引擎。但大多数专家系统都不能区分规则和元规则。所以必须对现有知识库中的元规则设定最高权限。当一条元规则触发时，能改变其他规则优先权的一些重要信息会被添加到数据库。

2.9 基于规则的专家系统的优点和缺点

当构建基于知识的系统时，基于规则的专家系统被公认为最好的选择。

什么特征使得基于规则的专家系统被知识工程师青睐

这些特征是：

- 自然语言表达。专家通常会使用这样的表达来解释解决问题的过程：“在什么—什么情况下，我如何—如何做。”这样的表达可以被很自然地表达为 IF-THEN 产生式规则。
- 统一结构。产生式规则具有统一的 IF-THEN 结构。每一条规则都是一个独立的知识。产生式规则的语法使得规则具有自释性。

- 知识与处理的相分离。基于规则的专家系统的结构为知识库和推理引擎提供了有效的分离机制。因此，能够使用同一个专家系统框架开发不同的应用，系统本身也容易扩展。在不干扰控制结构的同时通过添加一些规则，还能使系统更聪明。
- 处理不完整、不确定的知识。大多数基于规则的专家系统都能表达和推理不完整、不确定的知识。例如：

```
IF      season is autumn
AND    sky is 'cloudy'
AND    wind is low
THEN   forecast is clear    {cf 0.1};
        forecast is drizzle {cf 1.0};
        forecast is rain    {cf 0.9}
```

这条规则就表达了下面这条句子的不确定性。

“如果是秋季，看似在下毛毛雨，那么今天可能又很潮湿。”

这条规则用数值表达不确定性，称为确信因子 {cf 0.1}。专家系统使用确信因子来确立规则结论的可信度或者可信水平。我们将在第 3 章具体讨论这一话题。

这些特征使得专家系统在现实问题的知识表达上非常适用。

基于规则的专家系统就完美无缺了吗

其有 3 个主要的缺点：

- 规则之间的关系不透明。尽管单条规则都比较简单，也是自释性的，大量规则间的逻辑关系却可能不透明。在基于规则的系统中，难以观察单条规则如何对整个策略起作用，原因在于基于规则的专家系统缺乏分层的知识表达。
- 低效的搜索策略。推理引擎在每个周期中搜索所有的规则。当规则很多时（多于 100 条规则），系统速度会很慢。基于规则的大型系统可能就不适用于实时应用。
- 没有学习能力。一般的基于规则的专家系统都不具备从经验中学习的能力。人类专家知道何时打破规则，而专家系统并不能自动修改知识库，例如调整规则、添加规则。修改和维护系统的任务仍然由知识工程师来做。

2.10 小结

本章介绍了基于规则的专家系统，简要讨论了何为知识，专家如何以产生式规则形式表达知识。我们分析了专家系统开发团队的主要成员，描述了基于规则的系统的结构，分析了专家系统的主要特征，并指出专家系统也有出错的时候。接下来回顾了前向链接、后向链接推理技术，对冲突消解策略进行了讨论。最后，我们分析了基于规则的专家系统的优缺点。

本章的主要内容为：

- 知识是对主题的理论和实践两方面的理解，是当前所知的总和。
- 专家是在特定领域具有深厚知识（以事实和规则的形式体现）和丰富经验的人，能够做他人做不到的事情。
- 专家通常以产生式规则形式表达知识。
- 产生式规则用 IF（前提）THEN（结论）语句表达，它是最常用的知识表达方式。关系、建议、指示、策略以及启发式方法都可以用规则表达。
- 专家系统是在狭窄问题领域具有专家水平的计算机程序。基于规则的专家系统是最普及的专家系统。
- 开发基于规则的专家系统时，框架逐渐成为常用的选择。专家系统框架是没有知识的专家系统骨架。当针对应用构建系统时，用户只需以规则形式添加知识，并提供相关数据。因

此，专家系统框架大大降低了系统的开发时间。

- 专家系统团队由领域专家、知识工程师、程序员、项目经理以及终端用户组成。知识工程师设计、构建和测试专家系统，并从领域专家那里获取知识，确定推理方法，以及选择开发软件。当使用专家系统框架开发小型系统时，项目经理、知识工程师、程序员甚至领域专家都可能是同一个人。
- 基于规则的专家系统包括 5 个基本组成部分：知识库、数据库、推理引擎、解释设备以及用户界面。知识库包含用一系列规则表示的知识。数据库包含一系列事实，用于和规则的 IF 部分相匹配。推理引擎链接规则和事实，执行推理，以使系统得出解决方案。解释设备帮助用户查询系统如何得出一个结论，以及为什么需要某一个事实。用户界面则是用户和专家系统的互动途径。
- 专家系统将知识库和推理引擎分隔开，以分离知识和对知识的处理，从而使专家系统的构建和维护更加容易。当使用专家系统框架时，知识工程师或者专家只需将知识录入知识库。每添加一条新规则，都会扩充知识，也使专家系统更聪明。
- 专家系统在问题-解决会话中跟踪被触发的规则，从而具备一定的解释能力。
- 专家系统能够处理不完整、不确定的数据，也允许不精确的推理，这不同于传统程序。不过正如人类专家那样，当信息不完整或模糊时，专家系统也有出错的时候。
- 进行搜索和推理有两个主要方法：前向链接和后向链接推理技术。前向链接是数据驱动的：从已知数据出发进行推理，直到没有规则可以被激活。后向链接是目标驱动的：预先设定目标，推理引擎负责寻找证明目标的论据。
- 如果在一个循环内有多条规则满足触发条件，推理引擎必须决定该触发哪条规则，这种行为称为冲突消解。
- 基于规则的专家系统的优点是：能够有自然的知识表达方式，有统一的结构，知识和对知识的处理相分离，能处理不完整、不确定的知识。
- 基于规则的专家系统的缺点是：规则之间的关系不透明，搜索策略低效，不具备学习能力。

复习题

- 2.1 知识是什么？解释为什么在特定领域的有限范围内专家有丰富的知识。什么是启发式方法？
- 2.2 产生式规则是什么？举例说明产生式规则的两个必要部分。
- 2.3 列举并描述专家系统开发团队的 5 个主要成员。知识工程师的任务是什么？
- 2.4 什么是专家系统框架？解释为什么专家系统框架会大大降低系统的开发时间。
- 2.5 什么是产生式系统模型？列举并描述专家系统的 5 个基本组成部分。
- 2.6 专家系统的主要特征是什么？专家系统与传统程序的区别在哪里？
- 2.7 专家系统会出错吗？为什么？
- 2.8 描述前向链接推理技术并举例。
- 2.9 描述后向链接推理技术并举例。
- 2.10 列举前向链接推理技术适合解决的问题。为什么后向链接适合解决诊断问题？
- 2.11 什么是规则冲突集？怎么解决冲突？列举并描述基本的冲突消解方法。
- 2.12 列举基于规则的专家系统的优缺点。

参考文献

- Brachman, R.J. and Levesque, H.J. (2004). *Knowledge Representation and Reasoning*. Elsevier, San Francisco.
- Duda, R., Gaschnig, J. and Hart, P. (1979). Model design in the PROSPECTOR consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.
- Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.
- Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program, *Machine Intelligence 6*, B. Meltzer and D. Michie, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 165–190.
- Giarratano, J. and Riley, G. (2004). *Expert Systems: Principles and Programming*, 4th edn. Thomson/PWS Publishing Company, Boston, MA.
- Negnevitsky, M. (2008). Computational intelligence approach to crisis management in power systems, *International Journal of Automation and Control*, 2(2/3), 247–273.
- Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ.
- Shirai, Y. and Tsuji, J. (1985). *Artificial Intelligence: Concepts, Technologies and Applications*. John Wiley, New York.
- Shortliffe, E.H. (1976). *MYCIN: Computer-Based Medical Consultations*. Elsevier Press, New York.
- Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading, MA.
- Waterman, D.A. and Hayes-Roth, F. (1978). An overview of pattern-directed inference systems, *Pattern-Directed Inference Systems*, D.A. Waterman and F. Hayes-Roth, eds, Academic Press, New York.

基于规则的专家系统中的不确定性管理

本章主要介绍不确定性管理范例、贝叶斯推理和确信因子，讨论它们的优缺点，并以实例阐述这些理论。

3.1 不确定性简介

不完美是可供人类专家使用的信息的共同特点之一。信息可能是不完整的、不一致的、不确定的，也许会同时有这3个方面。也就是说，这样的信息总是不适合于解决问题。然而，专家却能应对这些缺陷，并通常做出正确的判断和决定。相应地，专家系统也应能够处理不确定因素，并给出有效的结论。

在专家系统中，不确定性指什么

不确定性是指，不具有使我们得出完美可信结论所需的准确的知识（Stephanou and Sage, 1987）。传统逻辑只允许精确推理，总假设存在完善知识，并总能使用排中律：

```
IF      A is true
THEN   A is not false
```

和

```
IF      B is false
THEN   B is not true
```

然而，用专家系统来解决的大多数实际问题并不能提供精确的知识，而是提供不精确、不完整甚至不可测的数据。

专家系统中不确定性知识的来源是什么

我们一般将来源分为4种：弱暗示、不精确的语言、未知数据，以及合并不同专家观点时的困难（Bonissone and Tong, 1985）。下面具体分析这4种来源。

- 弱暗示。基于规则的专家系统常面临弱暗示和模糊联系的问题。在确立规则的 IF（条件）和 THEN（行为）两部分的具体关联时，领域专家和知识工程师会感到棘手，或者难以进行。因此，处理模糊联系就成了专家系统的任务，例如用数值型确信因子表达关联度。
- 不精确的语言。自然语言是天生模糊、不精确的。我们在描述事实时，会用到“常常、有时、频繁地、几乎不”之类的术语。因此，难以用精确的 IF - THEN 产生式规则表达知识。如果将事实含义定量化，就可以用于专家系统。1944 年，Ray Simpson 让 355 个高中生或大学生对 20 个类似“常常”的术语表达为 1 ~ 100 范围内的数值（Simpson, 1944）。在 1968 年，Milton Hakel 重做了这个实验（Hakel, 1968）。结果如表 3.1 所示。通过量化术语含义，使专家系统能建立规则的 IF（条件）和事实之间的恰当匹配。
- 未知数据。当数据不完整或缺失时，唯一的方法是将值设为“未知”，以使推理继续下去。
- 融合不同专家的观点。大型专家系统通常会融合多个专家的知识 and 经验。例如，在开发勘探专家系统 PROSPECTOR 时，有 9 个专家参与（Duda et al., 1979）。不过，专家们的结论很少一致。观点不同导致规则冲突。为了解决冲突，知识工程师需要为每位专家分

配一个权重，并按权重综合各方结论。事实上，即使只有一个领域专家，他的经验水平也在变化。另外，不存在确定权重的系统性方法。

总之，由于任何实际领域都包含不确定知识，专家系统应能够管理不确定性，能够处理不完整、不一致甚至缺失的数据。在基于规则的专家系统中，已有许多数值型和非数值型方法来处理不确定性（Bhatnagar and Kanai, 1986）。本章将讨论最通用的不确定性管理范例：贝叶斯推理和确信因子。下面首先回顾一下经典概率论的基本原理。

表 3.1 模糊术语、不精确术语在时间频率范围上的量化

Ray Simpson (1944)		Milton Hakel (1968)	
术语	均值	术语	均值
Always	99	Always	100
Very often	88	Very often	87
Usually	85	Usually	79
Often	78	Often	74
Generally	78	Rather often	74
Frequently	73	Frequently	72
Rather often	65	Generally	72
About as often as not	50	About as often as not	50
Now and then	20	Now and then	34
Sometimes	20	Sometimes	29
Occasionally	20	Occasionally	28
Once in a while	15	Once in a while	22
Not often	13	Not often	16
Usually not	10	Usually not	16
Seldom	10	Seldom	9
Hardly ever	7	Hardly ever	8
Very seldom	6	Very seldom	7
Rarely	5	Rarely	5
Almost never	3	Almost never	2
Never	0	Never	0

3.2 概率论基本知识

概率这一基本概念在我们的日常生活中扮演着重要角色，例如要下雨的概率，得到提升的可能性，澳大利亚板球队在下一个赛季获胜的几率，中了 Tattslotto 彩票百万美元的概率等。

概率的概念有着很长的历史，可回溯到上千年之前，那时口语中已经出现了诸如 “probably”、“likely”、“maybe”、“perhaps”、“possibly” 之类的词（Good, 1959）。不过，概率的数学理论是 17 世纪才形成的。

如何定义概率

事件的概率是该事件发生所占的比例（Good, 1959）。概率也可以定义为可能性的科学度量。Feller（1966；1968）和 Fine（1973）写的两本著名教材中对现代概率理论进行了详细分析。本章仅研究在专家系统中与不确定性相关的基本概率概念。

概率可表示为 0（不可能发生）到 1（必然发生）范围内的数值型指标。大部分事件的概率

指标都严格限制在 0~1 之间，这意味着每一个事件至少有两个结果：有利的结果或成功，不利的结果或失败。

成功、失败的概率计算公式可表示为：

$$P(\text{成功}) = \frac{\text{成功次数}}{\text{可能结果的总数}} \tag{3.1}$$

$$P(\text{失败}) = \frac{\text{失败次数}}{\text{可能结果的总数}} \tag{3.2}$$

因此，如果用 s 表示成功次数， f 表示失败次数，则有：

$$P(\text{成功}) = p = \frac{s}{s+f} \tag{3.3}$$

$$P(\text{失败}) = q = \frac{f}{s+f} \tag{3.4}$$

和

$$p + q = 1 \tag{3.5}$$

现在，我们来看经典的抛硬币和掷骰子的例子。在抛硬币时，正面朝上和反面朝上的概率是一样的。在一次投掷中， $s=f=1$ ，因此正面（或反面）朝上的概率是 0.5。

再以掷骰子为例，分析单次投掷时得到 6 点的概率。假设我们的目标是 6 点，由于单次投掷时，仅有一种情况是 6 点，而有 5 种情况不是 6 点，所以 $s=1, f=5$ 。得到 6 点的概率是

$$p = \frac{1}{1+5} = 0.1666$$

结果不是 6 点的概率是

$$q = \frac{5}{1+5} = 0.8333$$

到现在为止，我们所关注的事件都是独立、互斥的（事件不可能同时发生）。在掷骰子的例子中，得到 6 点与得到 1 点这两个事件就是互斥的，因为不可能在一次投掷中同时得到 6 点和 1 点。同时，如果事件不独立，可能影响其他事件发生的可能。如果在掷骰子时，知道 1 点不会出现，我们分析单次投掷中得到 6 点的概率。虽然有 5 种情况不会出现 6 点，由于 1 点不会出现，就排除了 1 种情况，所以，

$$p = \frac{1}{1+(5-1)}$$

若 $A、B$ 是真实世界中的事件。假设有 $A、B$ 两个不互斥的事件，一个事件的发生条件依赖于另一个事件的发生。在事件 B 发生的基础上事件 A 发生的概率称为条件概率，用数学形式表示为 $p(A|B)$ ，其中的竖线“|”表示后面事件已发生。完整的概率表达式可理解为“在事件 B 已发生的情况下，事件 A 发生的条件概率”。

$$p(A|B) = \frac{A \text{ 和 } B \text{ 都发生的次数}}{B \text{ 发生的次数}} \tag{3.6}$$

A 和 B 都发生的概率，称为 A 和 B 的联合概率，它的数学表达形式是 $p(A \cap B)$ 。事件 B 发生的次数，或者 B 发生的概率，用 $p(B)$ 表示。那么有：

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \tag{3.7}$$

类似地，在事件 A 发生的前提下，事件 B 发生的条件概率是

$$p(B|A) = \frac{p(B \cap A)}{p(A)} \tag{3.8}$$

因此，

$$p(B \cap A) = p(B|A) \times p(A) \tag{3.9}$$

由于联合概率具有可交换性，则有：

$$p(A \cap B) = p(B \cap A)$$

所以，

$$p(A \cap B) = p(B|A) \times p(A) \tag{3.10}$$

将公式 (3.10) 代入公式 (3.7)，则有：

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B)} \tag{3.11}$$

其中：

$p(A|B)$ 是在事件 B 发生时，事件 A 发生的条件概率；

$p(B|A)$ 是在事件 A 发生时事件 B 发生的条件概率；

$p(A)$ 是事件 A 发生的概率；

$p(B)$ 是事件 B 发生的概率。

公式 (3.11) 就是贝叶斯规则，是 18 世纪英国数学家 Thomas Bayes 首先提出了该规则，以他的名字命名的。

前面所讲的条件概率的概念都假定事件 A 依赖事件 B 。贝叶斯规则可以扩展为事件 A 依赖多个互斥事件 B_1, B_2, \dots, B_n 的情况。可将公式 (3.7) 扩展为以下情况：

$$\begin{aligned} p(A \cap B_1) &= p(A|B_1) \times p(B_1) \\ p(A \cap B_2) &= p(A|B_2) \times p(B_2) \\ &\vdots \\ p(A \cap B_n) &= p(A|B_n) \times p(B_n) \end{aligned}$$

或合并为：

$$\sum_{i=1}^n p(A \cap B_i) = \sum_{i=1}^n p(A|B_i) \times p(B_i) \tag{3.12}$$

如果公式 (3.12) 包含了图 3.1 中的所有事件 B_i ，则有公式

$$\sum_{i=1}^n p(A \cap B_i) = p(A) \tag{3.13}$$

因此，公式 (3.12) 变为下面形式：

$$p(A) = \sum_{i=1}^n p(A|B_i) \times p(B_i) \tag{3.14}$$

如果事件 A 的发生依赖于两个互斥事件，即 B 发生或 B 不发生，公式 (3.14) 则变为下面形式：

$$p(A) = p(A|B) \times p(B) + p(A|\neg B) \times p(\neg B) \tag{3.15}$$

其中， \neg 表示逻辑运算符 NOT。

同样的，

$$p(B) = p(B|A) \times p(A) + p(B|\neg A) \times p(\neg A) \tag{3.16}$$

将公式 (3.16) 代入公式 (3.11)，得到下面的贝叶斯规则公式：

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B|A) \times p(A) + p(B|\neg A) \times p(\neg A)} \tag{3.17}$$

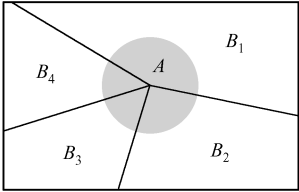


图 3.1 联合概率

公式 (3.17) 为专家系统中的不确定性管理提供了概率论的应用背景。

3.3 贝叶斯推理

有了公式 (3.17), 我们就可以暂时放下基本的概率理论, 将注意力转移到专家系统上。假设知识库中的所有规则都用下面形式表示:

IF E is true
THEN H is true {with probability p }

规则的含义是如果事件 E 发生, 则事件 H 发生的概率是 p 。

如果事件 E 已发生, 但不知道事件 H 是否也发生, 怎样计算 H 发生的概率

我们用公式 (3.17) 就能回答这个问题。我们现在使用符号 H 和 E , 而不是 A 和 B 。对于专家系统, H 表示假设, E 表示支持假设的论据。我们用 H 和 E 重写公式 (3.17):

$$p(H|E) = \frac{p(E|H) \times p(H)}{p(E|H) \times p(H) + p(E|\neg H) \times p(\neg H)} \quad (3.18)$$

其中:

$p(H)$ 是假设 H 为真的先验概率;

$p(E|H)$ 是假设 H 为真时导致论据 E 的概率;

$p(\neg H)$ 是假设 H 为假的先验概率;

$p(E|\neg H)$ 是在假设 H 为假时发现证据 E 的概率。

公式 (3.18) 表明在假设 H 的先验概率 $p(H)$ 必须在检验证据前定义。在专家系统中, 解决问题时需用到的概率由专家提供, 专家定义 $p(H)$ 和 $p(\neg H)$ 这两个先验概率, 以及条件概率 $p(E|H)$ 和 $p(E|\neg H)$ 。用户提供论据 E 的有关信息, 由专家系统计算 $p(H|E)$ 。 $p(H|E)$ 称为后验概率。

如果专家根据单个论据 E , 提出了多个假设 H_1, H_2, \dots, H_m , 而非一个假设, 或给定多个论据 E_1, E_2, \dots, E_n , 专家也提供了多个假设

可以将公式 (3.18) 推广到包含多个假设 H_1, H_2, \dots, H_m 和多个论据 E_1, E_2, \dots, E_n 的场合, 假设和论据都必须是互斥且完备的。

对于单个论据 E 和多个假设 H_1, H_2, \dots, H_m 的情况, 有下面公式:

$$p(H_i|E) = \frac{p(E|H_i) \times p(H_i)}{\sum_{k=1}^m p(E|H_k) \times p(H_k)} \quad (3.19)$$

对于多个论据 E_1, E_2, \dots, E_n 和多个假设 H_1, H_2, \dots, H_m 的情况, 则有:

$$p(H_i|E_1E_2\dots E_n) = \frac{p(E_1E_2\dots E_n|H_i) \times p(H_i)}{\sum_{k=1}^m p(E_1E_2\dots E_n|H_k) \times p(H_k)} \quad (3.20)$$

在应用公式 (3.20) 时, 需要论据和假设所有可能组合的条件概率。这对于专家来说是个巨大的负担, 实际上不可能完成。所以, 在专家系统中, 可忽略微小的论据, 并假设论据间条件独立 (Ng and Abramson, 1990)。因此我们得到了下面的公式

$$p(H_i|E_1E_2\dots E_n) = \frac{p(E_1|H_i) \times p(E_2|H_i) \times \dots \times p(E_n|H_i) \times p(H_i)}{\sum_{k=1}^m p(E_1|H_k) \times p(E_2|H_k) \times \dots \times p(E_n|H_k) \times p(H_k)} \quad (3.21)$$

专家系统怎样计算所有的后验概率, 并对真假设按可能性排名

考虑一个简单的例子。假设有 3 个条件独立的论据 E_1, E_2, E_3 , 让专家构造 3 个互斥且完备

的假设 H_1 、 H_2 、 H_3 ，并提供了这些假设的先验概率 $p(H_1)$ 、 $p(H_2)$ 、 $p(H_3)$ ，同时对于每一个假设，也确定了每一个论据的条件概率。表 3.2 为专家提供的先验概率和条件概率。

表 3.2 先验概率和条件概率

概率	假设		
	$i = 1$	$i = 2$	$i = 3$
$p(H_i)$	0.40	0.35	0.25
$p(E_1 H_i)$	0.3	0.8	0.5
$p(E_2 H_i)$	0.9	0.0	0.7
$p(E_3 H_i)$	0.6	0.7	0.9

假定我们首先看到了论据 E_3 ，专家系统按照公式 (3.19) 计算所有假设的后验概率：

$$p(H_i|E_3) = \frac{p(E_3|H_i) \times p(H_i)}{\sum_{k=1}^3 p(E_3|H_k) \times p(H_k)}, \quad i = 1, 2, 3$$

因此，

$$p(H_1|E_3) = \frac{0.6 \times 0.40}{0.6 \times 0.40 + 0.7 \times 0.35 + 0.9 \times 0.25} = 0.34$$

$$p(H_2|E_3) = \frac{0.7 \times 0.35}{0.6 \times 0.40 + 0.7 \times 0.35 + 0.9 \times 0.25} = 0.34$$

$$p(H_3|E_3) = \frac{0.9 \times 0.25}{0.6 \times 0.40 + 0.7 \times 0.35 + 0.9 \times 0.25} = 0.32$$

可以看出，当观察到论据 E_3 后，假设 H_1 的可信度降低，和 H_2 的可信度相同。假设 H_3 的可信度增加，几乎和 H_1 、 H_2 的可信度相等。

现在假设又得到了论据 E_1 ，用公式 (3.21) 计算后验概率：

$$p(H_i|E_1E_3) = \frac{p(E_1|H_i) \times p(E_3|H_i) \times p(H_i)}{\sum_{k=1}^3 p(E_1|H_k) \times p(E_3|H_k) \times p(H_k)}, \quad i = 1, 2, 3$$

所以有，

$$p(H_1|E_1E_3) = \frac{0.3 \times 0.6 \times 0.40}{0.3 \times 0.6 \times 0.40 + 0.8 \times 0.7 \times 0.35 + 0.5 \times 0.9 \times 0.25} = 0.19$$

$$p(H_2|E_1E_3) = \frac{0.8 \times 0.7 \times 0.35}{0.3 \times 0.6 \times 0.40 + 0.8 \times 0.7 \times 0.35 + 0.5 \times 0.9 \times 0.25} = 0.52$$

$$p(H_3|E_1E_3) = \frac{0.5 \times 0.9 \times 0.25}{0.3 \times 0.6 \times 0.40 + 0.8 \times 0.7 \times 0.35 + 0.5 \times 0.9 \times 0.25} = 0.29$$

现在假设 H_2 的可能性最大，而 H_1 的可能性急剧下降。

假设又观察到了论据 E_2 ，专家系统就可以计算所有假设最终的后验概率。

$$p(H_i|E_1E_2E_3) = \frac{p(E_1|H_i) \times p(E_2|H_i) \times p(E_3|H_i) \times p(H_i)}{\sum_{k=1}^3 p(E_1|H_k) \times p(E_2|H_k) \times p(E_3|H_k) \times p(H_k)}, \quad i = 1, 2, 3$$

因此，

$$p(H_1|E_1E_2E_3) = \frac{0.3 \times 0.9 \times 0.6 \times 0.40}{0.3 \times 0.9 \times 0.6 \times 0.40 + 0.8 \times 0.0 \times 0.7 \times 0.35 + 0.5 \times 0.7 \times 0.9 \times 0.25} = 0.45$$

$$\begin{aligned} p(H_2|E_1E_2E_3) &= \frac{0.8 \times 0.0 \times 0.7 \times 0.35}{0.3 \times 0.9 \times 0.6 \times 0.40 + 0.8 \times 0.0 \times 0.7 \times 0.35 + 0.5 \times 0.7 \times 0.9 \times 0.25} \\ &= 0 \\ p(H_3|E_1E_2E_3) &= \frac{0.5 \times 0.7 \times 0.9 \times 0.25}{0.3 \times 0.9 \times 0.6 \times 0.40 + 0.8 \times 0.0 \times 0.7 \times 0.35 + 0.5 \times 0.7 \times 0.9 \times 0.25} \\ &= 0.55 \end{aligned}$$

尽管专家按可能性对3个假设的最初排名是 H_1 、 H_2 、 H_3 ，在观察到3个论据（ E_1 、 E_2 、 E_3 ）后，只需考虑 H_1 、 H_3 ，可以放弃 H_2 。通过计算我们认为 H_3 的可能性要大于 H_1 。

用于勘探的专家系统 PROSPECTOR 是第一个使用贝叶斯规则计算 $p(H|E)$ 并在系统中传送不确定性的专家系统（Duda et al., 1979）。下面用一个简单的例子来解释专家系统中的贝叶斯推理。

3.4 FORECAST：论据累积的贝叶斯方法

现在我们开发一个解决实际问题（如天气预报）的专家系统。系统功能是预测明天是否下雨，它需要一些实际数据，这可以从气象局得到。

表3.3总结了伦敦在1982年3月的气象数据，包括每天的最低、最高温度，降雨量和日照时间。如果一天的降雨量为零，则说明一天不下雨。

专家系统所能提供的结果有两个：tomorrow is rain（明天下雨）、tomorrow is dry（明天不下雨），以及这两个结果的可能性。也就是说，专家系统必须给出“明天下雨”、“明天不下雨”这两个假设的条件概率。

在应用贝叶斯规则（公式（3.18））前，我们必须提供这两个假设的先验概率。

首先根据提供的数据写下两条基本规则，用它们预报明天的天气。

Rule: 1
IF today is rain
THEN tomorrow is rain

Rule: 2
IF today is dry
THEN tomorrow is dry

有了这两条规则，我们顶多犯10次错误，每次晴天后是雨天，雨天后是晴天。所以两个假设的先验概率都是0.5。接下来以下面形式重写规则：

Rule: 1
IF today is rain {LS 2.5 LN .6}
THEN tomorrow is rain {prior .5}

Rule: 2
IF today is dry {LS 1.6 LN .4}
THEN tomorrow is dry {prior .5}

表 3.3 1982 年 3 月伦敦天气总结

日	最低温度（℃）	最高温度（℃）	降雨量（mm）	日照（h）	实际天气	预报天气
1	9.4	11.0	17.5	3.2	下雨	—
2	4.2	12.5	4.1	6.2	下雨	下雨
3	7.6	11.2	7.7	1.1	下雨	下雨
4	5.7	10.5	0.0	4.3	晴天	下雨 ^①
5	3.0	12.0	0.0	9.5	晴天	晴天
6	4.4	9.6	0.0	3.5	晴天	晴天
7	4.8	9.4	4.6	10.1	下雨	下雨

(续)

日	最低温度 (℃)	最高温度 (℃)	降雨量 (mm)	日照 (h)	实际天气	预报天气
8	1.8	9.2	5.5	7.8	下雨	下雨
9	2.4	10.2	4.8	4.1	下雨	下雨
10	5.5	12.7	4.2	3.8	下雨	下雨
11	3.7	10.9	4.4	9.2	下雨	下雨
12	5.9	10.0	4.8	7.1	下雨	下雨
13	3.0	11.9	0.0	8.3	晴天	下雨 ^①
14	5.4	12.1	4.8	1.8	下雨	晴天 ^①
15	8.8	9.1	8.8	0.0	下雨	下雨
16	2.4	8.4	3.0	3.1	下雨	下雨
17	4.3	10.8	0.0	4.3	晴天	晴天
18	3.4	11.1	4.2	6.6	下雨	下雨
19	4.4	8.4	5.4	0.7	下雨	下雨
20	5.1	7.9	3.0	0.1	下雨	下雨
21	4.4	7.3	0.0	0.0	晴天	晴天
22	5.6	14.0	0.0	6.8	晴天	晴天
23	5.7	14.0	0.0	8.8	晴天	晴天
24	2.9	13.9	0.0	9.5	晴天	晴天
25	5.8	16.4	0.0	10.3	晴天	晴天
26	3.9	17.0	0.0	9.9	晴天	晴天
27	3.8	18.3	0.0	8.3	晴天	晴天
28	5.8	15.4	3.2	7.0	下雨	晴天 ^①
29	6.7	8.8	0.0	4.2	晴天	晴天
30	4.5	9.6	4.8	8.8	下雨	下雨
31	4.6	9.6	3.2	4.2	下雨	下雨

① 表示预报错误。

LS 的值是在论据 E 存在时, 专家估计假设 H 的可能性, 称为充分性似然值, 定义为 $p(E|H)$ 与 $p(E|\neg H)$ 的比值。

$$LS = \frac{p(E|H)}{p(E|\neg H)}$$

(3.22)

在我们的例子中, LS 是在明天下雨的情况下, 今天下雨的概率, 除以如果明天不下雨, 今天下雨的可能性,

$$LS = \frac{p(\text{今天下雨}|\text{明天下雨})}{p(\text{今天下雨}|\text{明天不下雨})}$$

LN 则是在没有论据 E 时, 假设 H 不被信任的度量, 也称为必要性似然值, 计算公式是

$$LN = \frac{p(\neg E|H)}{p(\neg E|\neg H)}$$

(3.23)

在我们的例子中, LN 表示在明天下雨的情况下今天不下雨的概率, 除以如果明天不下雨, 今天不下雨的概率,

$$LN = \frac{p(\text{今天不下雨}|\text{明天下雨})}{p(\text{今天不下雨}|\text{明天不下雨})}$$

注意, 从 LS 的计算公式不能推出 LN , 领域专家必须分别提供这两个值。

领域专家如何确定充分性似然值和必要性似然值呢？需要他们处理条件概率吗

为了确定 LS 和 LN 的值，专家不必给出精确的条件概率值，而是直接确定似然值。 LS 值大 ($LS \gg 1$) 就表明在论据存在时规则强烈支持假设， LN 值小 ($0 < LN < 1$) 表明在论据不存在时规则强烈反对假设。

由于从 LS 和 LN 的似然值可以轻易地算出条件概率，就可以使用贝叶斯规则来传送论据。

现在继续看伦敦天气的例子。规则 1 说明，如果今天下雨，明天也下雨的可能性就很大 ($LS = 2.5$)。即使今天不下雨，明天也可能有雨 ($LN = 0.6$)。

规则 2 则考虑了不下雨的情况。如果今天不下雨，明天也很有可能不下雨 ($LS = 1.6$)。而在今天下雨的情况下，明天下雨的概率要高于在今天不下雨的情况下，明天也不下雨的概率。为什么呢？ LS 和 LN 的值通常由领域专家给出。在天气预报的例子中， LS 和 LN 的值可以由气象局发布的统计信息来确认。规则 2 也给出了在今天下雨的情况下明天不下雨的可能性 ($LN = 0.4$)。

专家系统怎样得到明天不下雨或下雨的全部的概率

基于规则的专家系统将后项的先验概率 $p(H)$ 转换为先验几率。

$$O(H) = \frac{p(H)}{1 - p(H)} \quad (3.24)$$

仅在第一次调整后项的不确定性时，才使用先验概率。为了计算后验几率，当规则的前项（即论据）为真时，使用 LS 更新先验几率。当规则的前项假时，使用 LN 更新先验几率。

$$O(H|E) = LS \times O(H) \quad (3.25)$$

和

$$O(H|\neg E) = LN \times O(H) \quad (3.26)$$

再使用后验几率来恢复后验概率：

$$p(H|E) = \frac{O(H|E)}{1 + O(H|E)} \quad (3.27)$$

和

$$p(H|\neg E) = \frac{O(H|\neg E)}{1 + O(H|\neg E)} \quad (3.28)$$

用伦敦天气的例子解释这一方法的用法。假设用户指出今天下雨，规则 1 被触发，明天下雨的先验概率转换为先验几率：

$$O(\text{明天下雨}) = \frac{0.5}{1 - 0.5} = 1.0$$

今天下雨的论据将几率增加了 2.5 倍，所以明天下雨的概率从 0.5 增加到 0.71：

$$O(\text{明天下雨}|\text{今天下雨}) = 2.5 \times 1.0 = 2.5$$

$$p(\text{明天下雨}|\text{今天下雨}) = \frac{2.5}{1 + 2.5} = 0.71$$

规则 2 也被触发。明天不下雨的先验概率转换为先验几率，但今天下雨的论据将几率减小到 0.4。明天不下雨的概率就从 0.5 变为 0.29：

$$O(\text{明天不下雨}) = \frac{0.5}{1 - 0.5} = 1.0$$

$$O(\text{明天不下雨}|\text{今天下雨}) = 0.4 \times 1.0 = 0.4$$

$$p(\text{明天不下雨}|\text{今天下雨}) = \frac{0.4}{1 + 0.4} = 0.29$$

结论是，如果今天下雨，明天下雨的可能性是 71%，不下雨的可能性是 29%。
如果用户输入今天不下雨。同样的方式可计算出明天不下雨的可能性是 62%，下雨的可能性是 38%。

研究了论据的贝叶斯规则的基本原理后，我们可以向专家系统中添加新知识。首先我们需要确定天气发生变化的条件。通过对表 3.3 中的数据进行分析，我们建立了如下知识库（使用了专家系统框架 Leonardo）。

知识库

```
/* FORECAST: BAYESIAN ACCUMULATION OF EVIDENCE

control bayes

Rule: 1
if    today is rain {LS 2.5 LN .6}
then  tomorrow is rain {prior .5}

Rule: 2
if    today is dry {LS 1.6 LN .4}
then  tomorrow is dry {prior .5}

Rule: 3
if    today is rain
and   rainfall is low {LS 10 LN 1}
then  tomorrow is dry {prior .5}

Rule: 4
if    today is rain
and   rainfall is low
and   temperature is cold {LS 1.5 LN 1}
then  tomorrow is dry {prior .5}

Rule: 5
if    today is dry
and   temperature is warm {LS 2 LN .9}
then  tomorrow is rain {prior .5}

Rule: 6
if    today is dry
and   temperature is warm
and   sky is overcast {LS 5 LN 1}
then  tomorrow is rain {prior .5}

/* The SEEK directive sets up the goal of the rule set

seek tomorrow
```

对话

根据用户提供的信息，专家系统判断明天是否不下雨。用户提供的内容用箭头标出。如果降雨量少于 4.1mm，则为低降雨量。如果日平均气温等于或低于 7.0℃，则认为天气寒冷，如果高于 7.0℃，则认为天气暖和。如果日照时间低于 4.6h，则认为是阴天。

What is the weather today?
⇒ **rain**

```
Rule: 1
if    today is rain {LS 2.5 LN .6}
then  tomorrow is rain {prior .5}
```

$$O(\text{tomorrow is rain}) = \frac{0.5}{1 - 0.5} = 1.0$$
$$O(\text{tomorrow is rain} | \text{today is rain}) = 2.5 \times 1.0 = 2.5$$
$$p(\text{tomorrow is rain} | \text{today is rain}) = \frac{2.5}{1 + 2.5} = 0.71$$

tomorrow is rain {0.71}

Rule: 2

if today is dry {LS 1.6 LN .4}
then tomorrow is dry {prior .5}

$$O(\text{tomorrow is dry}) = \frac{0.5}{1 - 0.5} = 1.0$$

$$O(\text{tomorrow is dry} | \text{today is rain}) = 0.4 \times 1.0 = 0.4$$

$$p(\text{tomorrow is dry} | \text{today is rain}) = \frac{0.4}{1 + 0.4} = 0.29$$

tomorrow is rain {0.71}
 dry {0.29}

What is the rainfall today?

⇒ **low**

Rule: 3

if today is rain
and rainfall is low {LS 10 LN 1}
then tomorrow is dry {prior .5}

$$O(\text{tomorrow is dry}) = \frac{0.29}{1 - 0.29} = 0.41$$

$$O(\text{tomorrow is dry} | \text{today is rain} \cap \text{rainfall is low}) = 10 \times 0.41 = 4.1$$

$$p(\text{tomorrow is dry} | \text{today is rain} \cap \text{rainfall is low}) = \frac{4.1}{1 + 4.1} = 0.80$$

tomorrow is dry {0.80}
 rain {0.71}

What is the temperature today?

⇒ **cold**

Rule: 4

if today is rain
and rainfall is low
and temperature is cold {LS 1.5 LN 1}
then tomorrow is dry {prior .5}

$$O(\text{tomorrow is dry}) = \frac{0.80}{1 - 0.80} = 4$$

$$O(\text{tomorrow is dry} | \text{today is rain} \cap \text{rainfall is low} \cap \text{temperature is cold}) = 1.50 \times 4 = 6$$

$$p(\text{tomorrow is dry} | \text{today is rain} \cap \text{rainfall is low} \cap \text{temperature is cold}) = \frac{6}{1 + 6} = 0.86$$

tomorrow is dry {0.86}
 rain {0.71}

Rule: 5

if today is dry
and temperature is warm {LS 2 LN .9}
then tomorrow is rain {prior .5}

$$O(\text{tomorrow is rain}) = \frac{0.71}{1 - 0.71} = 2.45$$

$$O(\text{tomorrow is rain} | \text{today is not dry} \cap \text{temperature is not warm}) = 0.9 \times 2.45 = 2.21$$

$$p(\text{tomorrow is rain} | \text{today is not dry} \cap \text{temperature is not warm}) = \frac{2.21}{1 + 2.21} = 0.69$$

tomorrow is dry {0.86}
 rain {0.69}

What is the cloud cover today?

⇒ **overcast**


```
Rule: 6
if    today is dry
and   temperature is warm
and   sky is overcast {LS 5 LN 1}
then  tomorrow is rain {prior .5}

O(tomorrow is rain) =  $\frac{0.69}{1 - 0.69} = 2.23$ 

O(tomorrow is rain | today is not dry  $\cap$  temperature is not warm  $\cap$  sky is overcast)
= 1.0  $\times$  2.23 = 2.23

p(tomorrow is rain | today is not dry  $\cap$  temperature is not warm  $\cap$  sky is overcast)
=  $\frac{2.23}{1 + 2.23} = 0.69$ 

tomorrow is dry    {0.86}
rain               {0.69}
```

这说明两个假设“明天不下雨”、“明天下雨”都极有可能发生，但前者的可能性更高。

从表 3.3 可以看出专家系统仅犯过 4 次错误，成功率是 86%，比同样预测伦敦天气的 Naylor (1987) 结果还要好。

3.5 贝叶斯方法的偏差

贝叶斯推理框架需要概率值作为初始输入。对这些值的估计常常需要人的评判。不过心理研究表明，人给出的概率值要么与贝叶斯规则不一致，要么做得很糟糕 (Burns and Pearl, 1981; Tversky and Kahneman, 1982)。这说明条件概率可能与专家提供的先验概率不一致。例如，一辆汽车不能启动，在按下启动按钮时发出奇怪的噪声。如果汽车发出奇怪的噪声，故障出在启动按钮上的条件概率可表达为

```
IF    the symptom is 'odd noises'
THEN  the starter is bad {with probability 0.7}
```

而如果汽车发出奇怪的噪声，故障不在启动按钮上的条件概率是

$$p(\text{starter is not bad} | \text{odd noises}) = p(\text{starter is good} | \text{odd noises}) = 1 - 0.7 = 0.3$$

所以，我们有另外一条规则

```
IF    the symptom is 'odd noises'
THEN  the starter is good {with probability 0.3}
```

领域专家在处理条件概率时并不轻松，经常否定隐含概率的存在（在汽车例子中隐含概率是 0.3）。

在本例中，通过可用的统计信息和经验学习，可以得到下面两条规则：

```
IF    the starter is bad
THEN  the symptom is 'odd noises' {with probability 0.85}

IF    the starter is bad
THEN  the symptom is not 'odd noises' {with probability 0.15}
```

要使用贝叶斯规则，还必须要有先验概率，即当汽车不能启动时，启动按钮有故障的概率。这需要专家来提供。假设专家认为这一概率是 5%，我们应用贝叶斯规则得到下面的式子：

$$p(\text{starter is bad} | \text{odd noises}) = \frac{0.85 \times 0.05}{0.85 \times 0.05 + 0.15 \times 0.95} = 0.23$$

这个值远低于最初由专家提供的估计值 0.7。

为什么会不一致？专家错了吗

最明显的原因是专家在估计条件概率和先验概率时做了不同的假设。我们试着从后验概率 $p(\text{starter is bad} | \text{odd noises})$ 倒推先验概率 $p(\text{starter is bad})$ 。在本例中，可以假设

$$p(\text{starter is good}) = 1 - p(\text{starter is bad})$$

根据公式 (3.18)，我们得出

$$p(H) = \frac{p(H|E) \times p(E|\neg H)}{p(H|E) \times p(E|\neg H) + p(E|H) [1 - p(H|E)]}$$

其中：

- $p(H) = p(\text{starter is bad})$;
- $p(H|E) = p(\text{starter is bad} | \text{odd noises})$;
- $p(E|H) = p(\text{odd noises} | \text{starter is bad})$;
- $p(E|\neg H) = p(\text{odd noises} | \text{starter is good})$ 。

如果我们使用专家提供的 $p(\text{starter is bad} | \text{odd noises})$ 的值 0.7 作为正确值， $p(\text{starter is bad})$ 的先验概率应该是

$$p(H) = \frac{0.7 \times 0.15}{0.7 \times 0.15 + 0.85 \times (1 - 0.7)} = 0.29$$

这个值几乎是专家提供的值 5% 的 6 倍。因此专家应该对先验概率和条件概率分别进行不同的估计。

事实上，由专家提供的先验概率也会与充分性似然值 LS 、必要性似然值 LN 不一致。已有若干方法处理这一问题（Duda et al.，1976）。最常用的技术是分段线性插值模型（Duda et al.，1979），首次应用在 PROSPECTOR 中。

要使用贝叶斯规则，我们必须满足多个假定条件，如针对假设和逆假设，论据间都应是条件独立的。由于很少有实际问题满足这些假定条件，使用贝叶斯推理的系统并不多，用于矿产勘探的专家系统 PROSPECTOR 是其中之一（Duda et al.，1979）。

3.6 确信因子理论和基于论据的推理

确信因子理论是常用的替代贝叶斯推理的方法，这一理论的基本原理首次出现在 MYCIN 中，这是一个用于诊断、治疗传染性血液病和脑膜炎的专家系统（Shortliffe and Buchanan，1975）。MYCIN 的开发者发现，医学专家通常用既不符合逻辑一致性或数学一致性的方式表达术语的可信度。此外，不存在问题领域相关的可信赖的统计数据。所以，MYCIN 团队不能使用传统概率方法。于是，他们决定用确信因子（certainty factor, cf）来衡量专家对术语的可信度。确信因子的最大值是 +1.0（真），最小值是 -1.0（假）。正值表示可信度，负值表示不可信度。例如，如果专家说一个论据几乎是真的，将对 cf 值赋值 0.8。表 3.4 是 MYCIN 系统中的一些不确定性术语（Durkin，1994）。

表 3.4 不确定性术语及其解释

术语	确信因子
Definitely not	-1.0
Almost certainly not	-0.8
Probably not	-0.6
Maybe not	-0.4
Unknown	-0.2 ~ +0.2
Maybe	+0.4
Probably	+0.6
Almost certainly	+0.8
Definitely	+1.0

在使用确信因子的专家系统中，知识库中规则的语法是：

```
IF      <evidence>
THEN  <hypothesis> {cf}
```

其中，*cf* 是论据 *E* 发生时假设 *H* 的可信度。

可信度理论基于两个函数：可信度的度量 *MB* (*H*, *E*)，不可信度的度量 *MD* (*H*, *E*) (Shortliffe and Buchanan, 1975)。这两个函数分别表示在论据 *E* 出现时，假设 *H* 的可信度的增加程度和不可信度的增加程度。

可信度和不可信度可通过先验概率、条件概率计算 (Ng and Abramson, 1990)：

$$MB(H,E) = \begin{cases} 1 & \text{若 } p(H) = 1 \\ \frac{\max[p(H|E), p(H)] - p(H)}{\max[1, 0] - p(H)} & \text{否则} \end{cases} \tag{3.29}$$

$$MD(H,E) = \begin{cases} 1 & \text{若 } p(H) = 1 \\ \frac{\min[p(H|E), p(H)] - p(H)}{\min[1, 0] - p(H)} & \text{否则} \end{cases} \tag{3.30}$$

其中：

- p*(*H*) 是假设 *H* 为真的先验概率；
- p*(*H* | *E*) 是给定论据 *E* 时假设 *H* 为真的概率。

MB(*H*,*E*)和 *MD*(*H*,*E*)的取值范围都介于 0 ~ 1 之间。假设 *H* 的可信度和不可信度都依赖论据 *E* 的类型。有些事实可能会提高可信度,而有些事实可能会提高不可信度。

如何确定假设的可信度和不可信度的总体情况

使用下面的公式将可信度和不可信度综合为一个数值,即确信因子：

$$cf = \frac{MB(H,E) - MD(H,E)}{1 - \min[MB(H,E), MD(H,E)]} \tag{3.31}$$

因此，*cf* 表达了假设 *H* 的总体可信度，它在 MYCIN 中的范围是 -1 ~ +1。

可以通过一个例子来说明 MYCIN 中的方法。考虑下面一条简单的规则：

```
IF      A is X
THEN  B is Y
```

专家通常不能保证这个规则绝对有效。在一些情况下，即使规则的 IF 部分条件成立，即对象 *A* 的值是 *X*，对象 *B* 也可能有不同的 *Z* 值。也就是说，我们讨论的是准统计的不确定性。

当 *A* 取值 *X* 时，专家通常会将 *B* 的每一个可能值关联上一个确信因子。因此，规则应该像下面的形式：

```
IF      A is X
THEN  B is Y {cf 0.7};
      B is Z {cf 0.2}
```

另外 10% 哪去了

这个规则的意思是，假定 *A* 的值是 *X*，则 *B* 取 *Y* 值的概率是 70%，取 *Z* 值的概率是 20%，取其他值的概率是另外 10%。就是专家保留了对对象 *B* 取 *Y* 和 *Z* 之外其他取值的可能性。所以，对象 *B* 被赋予了多个值。

规则中的确信因子通过推理链来传送。确信因子的传送还包括当规则前项中的论据不确定时，确立规则后项的净确信度。单个规则前项的净确信度 *cf*(*H*, *E*)，可以由规则前项的确信因子 *cf*(*E*) 乘以规则的确信因子 *cf* 得到

$$cf(H,E) = cf(E) \times cf \tag{3.32}$$

例如，

IF the sky is clear
THEN the forecast is sunny {cf 0.8}

“sky is clear” 的当前确信因子是 0.5，那么

$$cf(H, E) = 0.5 \times 0.8 = 0.4$$

按照表 3.4，这个结果意味着“可能晴天”。

对于有多个前项的规则，专家系统如何计算确信因子

对于合取规则，例如：

IF <evidence E_1 >
AND <evidence E_2 >
:
AND <evidence E_n >
THEN <hypothesis H > {cf}

后项的净确信度，即假设 H 的确信度用以下公式计算：

$$cf(H, E_1 \cap E_2 \cap \dots \cap E_n) = \min[cf(E_1), cf(E_2), \dots, cf(E_n)] \times cf \quad (3.33)$$

例如，

IF sky is clear
AND the forecast is sunny
THEN the action is ‘wear sunglasses’ {cf 0.8}

“sky is clear” 的确信度是 0.9，“the forecast is sunny” 的确信度是 0.7，那么

$$cf(H, E_1 \cap E_2) = \min[0.9, 0.7] \times 0.8 = 0.7 \times 0.8 = 0.56$$

根据表 3.4，结论可以解释为“或许今天适合戴太阳镜”。

对于析取规则，如

IF <evidence E_1 >
OR <evidence E_2 >
:
OR <evidence E_n >
THEN <hypothesis H > {cf}

假设 H 的确信度通过下式计算：

$$cf(H, E_1 \cup E_2 \cup \dots \cup E_n) = \max[cf(E_1), cf(E_2), \dots, cf(E_n)] \times cf \quad (3.34)$$

例如，

IF sky is overcast
OR the forecast is rain
THEN the action is ‘take an umbrella’ {cf 0.9}

“sky is overcast” 的确信度是 0.6，“forecast is rain” 的确信度是 0.8，则

$$cf(H, E_1 \cup E_2) = \max[0.6, 0.8] \times 0.9 = 0.8 \times 0.9 = 0.72$$

结论可解释为“今天几乎肯定是要带上雨伞”。

有时两个或更多的规则会影响到同一个假设，这种情况下专家系统怎么办

当同一个结论要由两个或多个规则共同推出时，必须合并它们的确信因子，以得出假设的确信因子。假设知识库中有如下规则：

Rule 1: IF A is X
THEN C is Z {cf 0.8}

Rule 2: IF B is Y
THEN C is Z {cf 0.6}

如果规则 1 和规则 2 都被触发了，对象 C 取 Z 值的的确信度如何计算？按照常识，如果有来自不同来源（规则 1 和规则 2）的两个论据（ A is X 和 B is Y ），都支持同一个假设（ C is Z ），假设的可信度应该提高，应该比只有一个论据时更强。

我们使用下面的公式来计算结合后的确信因子（Durkin，1994）：

$$cf(cf_1, cf_2) = \begin{cases} cf_1 + cf_2 \times (1 - cf_1) & \text{若 } cf_1 > 0, cf_2 > 0 \\ \frac{cf_1 + cf_2}{1 - \min[|cf_1|, |cf_2|]} & \text{若 } cf_1 < 0 \text{ 或者 } cf_2 < 0 \\ cf_1 + cf_2 \times (1 + cf_1) & \text{若 } cf_1 < 0 \text{ 或者 } cf_2 < 0 \end{cases} \quad (3.35)$$

其中：

- cf_1 是规则 1 对假设 H 的确信度；
 - cf_2 是规则 2 对假设 H 的确信度；
 - $|cf_1|$ 和 $|cf_2|$ 分别是 cf_1 和 cf_2 的绝对值。
- 因此，我们假设

$$cf(E_1) = cf(E_2) = 1.0$$

由公式 (3.32) 可得

$$\begin{aligned} cf_1(H, E_1) &= cf(E_1) \times cf_1 = 1.0 \times 0.8 = 0.8 \\ cf_2(H, E_2) &= cf(E_2) \times cf_2 = 1.0 \times 0.6 = 0.6 \end{aligned}$$

由公式 (3.35) 可得

$$\begin{aligned} cf(cf_1, cf_2) &= cf_1(H, E_1) + cf_2(H, E_2) \times [1 - cf_1(H, E_1)] \\ &= 0.8 + 0.6 \times (1 - 0.8) = 0.92 \end{aligned}$$

这个例子表明假设的可信度增加，也验证了我们的预期。

接下来考虑规则的确信因子是负值的情况。假设

$$cf(E_1) = 1, cf(E_2) = -1.0$$

那么

$$\begin{aligned} cf_1(H, E_1) &= 1.0 \times 0.8 = 0.8 \\ cf_2(H, E_2) &= -1.0 \times 0.6 = -0.6 \end{aligned}$$

由公式 (3.35) 可得

$$cf(cf_1, cf_2) = \frac{cf_1(H, E_1) + cf_2(H, E_2)}{1 - \min[|cf_1(H, E_1)|, |cf_2(H, E_2)|]} = \frac{0.8 - 0.6}{1 - \min[0.8, 0.6]} = 0.5$$

该例说明了当一条规则支持假设而另一条规则反对假设的情况下，如何计算确信因子，即净可信度。

如果规则的确信因子都是负值呢？假设

$$cf(E_1) = cf(E_2) = -1.0$$

那么

$$\begin{aligned} cf_1(H, E_1) &= -1.0 \times 0.8 = -0.8 \\ cf_2(H, E_2) &= -1.0 \times 0.6 = -0.6 \end{aligned}$$

由公式 (3.35) 可得：

$$\begin{aligned} cf(cf_1, cf_2) &= cf_1(H, E_1) + cf_2(H, E_2) \times [1 + cf_1(H, E_1)] \\ &= -0.8 - 0.6 \times (1 - 0.8) = -0.92 \end{aligned}$$

这个例子中假设的不可信度有了增加。

确信因子理论是贝叶斯推理的实用的替代方法。合并确信因子的启发式方法不同于合并概率的方法。确定理论不是“纯粹数学”的方法，而是模拟人类专家思考过程的方法。

下面通过 3.4 节介绍的专家系统 FORECAST，来说明基于论据的推理以及确信因子在一组规则中传送的方法。

3.7 FORECAST：确信因子的应用

这个专家系统的任务是预测明天是否下雨，即为多值对象 tomorrow 建立确信因子。为了简化任务，我们仍使用 3.4 节的相关规则。

知识库

```
/* FORECAST: AN APPLICATION OF CERTAINTY FACTORS
```

```
control cf
```

```
control 'threshold 0.01'
```

```
Rule: 1
```

```
if    today is rain
```

```
then tomorrow is rain {cf 0.5}
```

```
Rule: 2
```

```
if    today is dry
```

```
then tomorrow is dry {cf 0.5}
```

```
Rule: 3
```

```
if    today is rain
```

```
and  rainfall is low
```

```
then tomorrow is dry {cf 0.6}
```

```
Rule: 4
```

```
if    today is rain
```

```
and  rainfall is low
```

```
and  temperature is cold
```

```
then tomorrow is dry {cf 0.7}
```

```
Rule: 5
```

```
if    today is dry
```

```
and  temperature is warm
```

```
then tomorrow is rain {cf 0.65}
```

```
Rule: 6
```

```
if    today is dry
```

```
and  temperature is warm
```

```
and  sky is overcast
```

```
then tomorrow is rain {cf 0.55}
```

```
seek tomorrow
```

对话

为了应用基于确信因子的不精确推理技术，专家系统让用户既输入对象的值，又输入与值关联的确信度。例如，如果使用 Leonardo 规定的 0 ~ 1 的范围，可能会出现下面的对话：

```
What is the weather today?
```

```
⇒ rain
```

```
Rule: 1
```

```
if    today is rain
```

```
then tomorrow is rain {cf 0.5}
```

```
cf(tomorrow is rain, today is rain) = cf(today is rain) × cf = 1.0 × 0.5 = 0.5
```

```
tomorrow is rain    {0.50}
```

```
What is the rainfall today?
```

```
⇒ low
```

```
To what degree do you believe the rainfall is low? Enter a numeric certainty
between 0 and 1.0 inclusive.
```

```
⇒ 0.8
```

```
Rule: 3
```

```
if    today is rain
```

```
and  rainfall is low
```

```
then tomorrow is dry {cf 0.6}
```

$cf(\text{tomorrow is dry, today is rain} \cap \text{rainfall is low})$
 $= \min [cf(\text{today is rain}), cf(\text{rainfall is low})] \times cf = \min [1, 0.8] \times 0.6 = 0.48$

tomorrow is rain {0.50}
 dry {0.48}

What is the temperature today?
 \Rightarrow **cold**

To what degree do you believe the temperature is cold? Enter a numeric certainty between 0 and 1.0 inclusive.
 \Rightarrow **0.9**

Rule: 4
 if today is rain
 and rainfall is low
 and temperature is cold
 then tomorrow is dry {cf 0.7}

$cf(\text{tomorrow is dry, today is rain} \cap \text{rainfall is low} \cap \text{temperature is cold})$
 $= \min [cf(\text{today is rain}), cf(\text{rainfall is low}), cf(\text{temperature is cold})] \times cf$
 $= \min [1, 0.8, 0.9] \times 0.7 = 0.56$

tomorrow is dry {0.56}
 rain {0.50}

$cf(cf_{\text{Rule:3}}, cf_{\text{Rule:4}}) = cf_{\text{Rule:3}} + cf_{\text{Rule:4}} \times (1 - cf_{\text{Rule:3}})$
 $= 0.48 + 0.56 \times (1 - 0.48) = 0.77$

tomorrow is dry {0.77}
 rain {0.50}

3.8 贝叶斯推理和确信因子的对比

在上一节，我们介绍了专家系统中的两个流行的不确定性管理技术。现在我们比较这两个技术，分析它们都适合哪些问题。

概率理论是处理不精确知识和随机数据的最古老、最好的技术，很适用于预测和计划相关的问题，因为可以获得统计数据，并编写准确的概率语句。

专家系统 PROSPECTOR 是一个帮助勘探地质学家搜寻矿床的专家系统，它使用了贝叶斯技术。这个专家系统开发得很成功，如通过使用地质、地球物理、地球化学数据，PROSPECTOR 预测出了华盛顿州 Tolman 山附近的一个钼矿床（Campbell et al.，1982）。PROSPECTOR 团队能利用已知矿床的有效数据和统计信息，也定义了每一个事件的概率。PROSPECTOR 团队也假定了论据之间具有条件独立性，这也是使用贝叶斯规则的前提。

但是，在许多应用领域，并不总能得到可用的统计信息，或总能假定论据间满足条件独立性。所以，许多研究者发现贝叶斯方法对他们的工作并不适合。例如，在 MYCIN 中，由于医学领域通常不提供需要的数据，Shortliffe 和 Buchanan 就不能使用传统的概率方法（Shortliffe and Buchanan，1975）。

尽管确信因子方法缺乏概率论那样的数学正确性，在一些领域如在诊断领域、尤其是医药领域，它还是能够胜过贝叶斯推理方法的。在像 MYCIN 这样的诊断专家系统中，专家根据自己的知识或直觉判断来确定规则和确信因子。在概率未知、或难以获取或代价昂贵的场合，可以使用确信因子。论据推理机制可以增量管理获得的论据、合取式假设、析取式假设，以及可信度不同的论据。另外，确信因子方法对基于规则的专家系统中的控制流能提供更好的解释。

贝叶斯方法和确信因子虽彼此不同，它们却有着共同问题：找到能够量化个人的、主观的、定性的信息的专家。人类很容易有偏差，所以选择哪一个不确定性管理技术很大程度上依赖于

现有的领域专家。

当存在可靠的统计数据，知识工程师能够主持大局，也有能参与严密的决策分析对话的专家时，贝叶斯方法应该是最适合的。这3个条件缺少任何一个，使用贝叶斯方法就显得太武断，也会由于偏差而不能得到有价值的结果。还需提及的是，贝叶斯信念传播的复杂度是指数级的，不适用于大型知识库。

虽然缺乏形式基础，但是确信因子技术为专家系统处理不确定性提供了一个简单的方法，在许多应用中的结果也都是可以接受的。

3.9 小结

本章介绍了用于专家系统的两个不确定性管理技术：贝叶斯推理和确信因子。分析了不确定知识的主要来源，简要回顾了概率理论。我们讨论了可累积论据的贝叶斯方法，并基于贝叶斯方法开发了一个简单的专家系统。之后介绍了确信因子理论（贝叶斯推理的常用替代方法），并基于论据推理构建了一个专家系统。最后，将贝叶斯推理和确信因子进行了对比，分析了它们的适用范围。

本章的主要内容如下：

- 不确定性是由于缺乏帮助我们得出完美可靠结论的精确知识。专家系统中的不确定知识的主要来源是：弱暗示、不精确的语言、数据缺失、综合不同专家的观点。
- 概率理论为专家系统中的不确定性管理提供了一个精确的、数学校正的方法。在给定论据的前提下，我们可使用贝叶斯规则计算出一个假设的概率。
- 用于矿床勘探的专家系统 PROSPECTOR 是第一个成功应用贝叶斯规则传送不确定性的系统。
- 要使用贝叶斯方法，专家需要提供假设 H 的先验概率、充分性似然值 LS （衡量在给定论据 E 时假设的可信度）、必要性似然值（衡量在给定论据 E 时假设的不可信度）。贝叶斯方法使用下面的规则形式：

```
IF      E is true {LS, LN}
THEN   H is true {prior probability}
```

- 要想使用贝叶斯方法，论据间必须满足条件独立，还应该有可靠的统计数据，并为每个假设定义先验概率。在实际问题上，往往不能满足这些要求，所以只有少数系统可以使用贝叶斯推理。
- 确信因子理论是贝叶斯方法的常用替代方法。这一理论的基本原理首次出现在诊断医疗专家系统 MYCIN 中。
- 确信因子理论为专家系统中的不确定管理提供了一个判断方法。使用这一方法时，需要专家提供确信因子 cf 的值，用来表示给定论据 E 时假设 H 的可信度水平。确信因子方法使用如下规则：

```
IF      E is true
THEN   H is true {cf}
```

- 当概率未知或不易获得时，会使用确信因子。确定理论可以增量获得的管理论据、合取式假设、析取式假设，以及具有不同可信度的论据。
- 贝叶斯推理和确定理论面临的共同问题是：需要一个能够量化主观信息和定性信息的专家。

复习题

1. 什么是不确定性？什么情况下知识可以不精确，数据可以不完整、不一致？举例说明不精确的知识。

2. 什么是概率？用数学方式描述在事件 B 发生时事件 A 发生的条件概率。什么是贝叶斯规则？
3. 什么是贝叶斯推理？专家系统怎样对可能的真假设进行排序？举例说明。
4. 为什么 PROSPECTOR 团队能够将贝叶斯方法作为不确定性管理技术？使用贝叶斯推理必须要具备哪些条件？
5. 什么是充分性似然值和必要性似然值？专家如何确定 LS 和 LN 的值？
6. 什么是先验概率？举例说明基于贝叶斯推理的专家系统的规则表达方式。
7. 基于规则的专家系统如何使用贝叶斯方法传送不确定性？
8. 条件概率为什么也许会与专家提供的先验概率不一致？举例说明。
9. 为什么确信因子理论可被看做贝叶斯推理的实用替代方法？什么是可信度和不可信度的度量？定义一个确信因子。
10. 专家系统如何确定“与”、“或”规则的净确定性？举例说明。
11. 专家系统如何合并影响同一个假设的多个规则的确信因子？举例说明。
12. 请比较贝叶斯推理和确信因子。它们分别适合什么样的应用场景？为什么？这两个方法的共同问题是什么？

参考文献

- Bhatnagar, R.K. and Kanal, L.N. (1986). Handling uncertain information: a review of numeric and non-numeric methods, *Uncertainty in AI*, L.N. Kanal and J.F. Lemmer, eds, Elsevier North-Holland, New York, pp. 3–26.
- Bonissone, P.P. and Tong, R.M. (1985). Reasoning with uncertainty in expert systems, *International Journal on Man–Machine Studies*, 22(3), 241–250.
- Burns, M. and Pearl, J. (1981). Causal and diagnostic inferences: a comparison of validity, *Organizational Behaviour and Human Performance*, 28, 379–394.
- Campbell, A.N., Hollister, V.F., Duda, R.O. and Hart, P.E. (1982). Recognition of a hidden mineral deposit by an artificial intelligence program, *Science*, 217(3), 927–929.
- Duda, R.O., Hart, P.E. and Nilsson, N.L. (1976). Subjective Bayesian methods for a rule-based inference system, *Proceedings of the National Computer Conference (AFIPS)*, vol. 45, pp. 1075–1082.
- Duda, R.O., Gaschnig, J. and Hart, P.E. (1979). Model design in the PROSPECTOR consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.
- Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.
- Feller, W. (1966). *An Introduction to Probability Theory and its Applications*, vol. 2, John Wiley, New York.
- Feller, W. (1968). *An Introduction to Probability Theory and its Applications*, vol. 1, 3rd edn. John Wiley, New York.
- Fine, T.L. (1973). *Theories of Probability: An Examination of Foundations*. Academic Press, New York.
- Firebaugh, M.W. (1989). *Artificial Intelligence: A Knowledge-based Approach*. PWS-KENT Publishing Company, Boston, MA.
- Good, I.J. (1959). Kinds of probability, *Science*, 129(3347), 443–447.
- Hakel, M.D. (1968). How often is often? *American Psychologist*, no. 23, 533–534.
- Naylor, C. (1987). *Build Your Own Expert System*. Sigma Press, Wilmslow, Cheshire.
- Ng, K.-C. and Abramson, B. (1990). Uncertainty management in expert systems, *IEEE Expert*, 5(2), 29–47.

- Shortliffe, E.H. and Buchanan, B.G. (1975). A model of inexact reasoning in medicine, *Mathematical Biosciences*, 23(3/4), 351–379.
- Simpson, R. (1944). The specific meanings of certain terms indicating differing degrees of frequency, *The Quarterly Journal of Speech*, no. 30, 328–330.
- Stephanou, H.E. and Sage, A.P. (1987). Perspectives on imperfect information processing, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(5), 780–798.
- Tversky, A. and Kahneman, D. (1982). Causal schemes in judgements under uncertainty, *Judgements Under Uncertainty: Heuristics and Biases*, D. Kahneman, P. Slovic and A. Tversky, eds, Cambridge University Press, New York.



人工智能 智能系统指南 (原书第3版)

Artificial Intelligence A Guide to Intelligent Systems Third Edition

人工智能经常被人们认为是计算机科学中一门高度复杂甚至令人生畏的学科。长期以来人工智能方面的书籍往往包含复杂的矩阵代数和微分方程。本书基于作者多年面向微积分知识甚微的学生授课时所用的讲义,假定读者没有编程经验,以简单易懂的方式介绍了智能系统的基础知识。

本书目前已经被国际上多所大学(例如,德国的马格德堡大学、日本的广岛大学、美国的波士顿大学和罗切斯特理工学院等)采纳为教材。

如果您正在寻找关于人工智能或智能系统设计课程的浅显易懂的入门级教材,如果您不是计算机科学领域的专业人员而又正在寻找介绍基于知识系统最新技术发展的自学指南,本书将是您的最佳选择。

本书的主要内容: 基于规则的专家系统、不确定性管理技术、模糊专家系统、基于框架的专家系统、人工神经网络、进化计算、混合智能系统、知识工程、数据挖掘。

本版的新增内容: 与上一版相比,本版进行了全面更新,以反映人工智能领域的最新进展。其中新增了“数据挖掘与知识发现”一章和“自组织神经网络”及聚类相关内容,同时补充了4个新的案例研究。

作者简介

Michael Negnevitsky 澳大利亚塔斯马尼亚大学电气工程和计算机科学系教授。他的许多研究课题都涉及人工智能和软计算。他一直致力于电气工程、过程控制和环境工程中智能系统的开发和应用,发表了300多篇论文,著有2本专著,并获得了4项发明专利。



影印版

书号: 978-7-111-35822-0

定价: 49.00元

客服热线: (010) 88378991, 88361066
购书热线: (010) 68326294, 88379649, 68995259
投稿热线: (010) 88379604
读者信箱: hzsj@hzbook.com

华章网站 <http://www.hzbook.com>

网上购书: www.china-pub.com

封面设计: 包昂 杨彬

www.aibbt.com 让未来触手可及

上架指导: 计算机/人工智能

ISBN 978-7-111-38455-7



9 787111 384557

定价: 49.00元